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

CONVERSION OF WHISPERED TO NORMAL VOICE

KONVERZE ŠEPTANÉ ŘEČI NA NORMÁLNÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **RICHARD GAJDA**
AUTOR PRÁCE

SUPERVISOR **Ing. JAN BRUKNER**
VEDOUCÍ PRÁCE

BRNO 2020

Brno University of Technology
Faculty of Information Technology

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

Bachelor's Thesis Specification

22505

Student: **Gajda Richard**
Programme: Information Technology
Title: **Conversion of Whispered to Normal Voice**
Category: Speech and Natural Language Processing

Assignment:

1. Get acquainted with techniques for speech parametrization and re-synthesis.
2. Study modern approaches for whispered to normal speech conversion based on machine learning.
3. Acquire or collect dataset of parallel whispered and normal speech for training and evaluation.
4. Implement an existing system using available signal processing and machine learning libraries and evaluate the results.
5. Implement a mobile application utilizing the used method.
6. Create a short video or poster presenting your work.

Recommended literature:

- According to supervisor's advice

Requirements for the first semester:

- Items 1 to 4 of the assignment, start working on item 5.

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

Supervisor: **Brukner Jan, Ing.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2020
Submission deadline: May 12, 2021
Approval date: May 18, 2021

Abstract

The aim of this thesis is to develop a working program, that converts whispered speech input into voice using vocal excitation prediction, which is obtained from a neural network. The work is based on a study from Indian Institute of Science in Bengalore, India. The approach to the solution is the following: to acquire a dataset from training speakers, to implement the speech parameterization using the WORLD vocoder, to implement and train the neural networks, to experiment, to evaluate the results and, finally, to propose future applications and improvements.

Abstrakt

Cílem této práce je vyvinout funkční program, který konvertuje vstupní šeptanou řeč na neutrální za pomoci predikce hlasového buzení, která je získána pomocí neuronových sítí. Práce je založena na studii z Indian Institute of Science v indickém Bengalúru. Řešení je provedeno následovně: nejprve získáme trénovací dataset řečníků, poté implementujeme zpracování řeči a její parametrizaci za pomoci vokodéru WORLD, vytvoříme a natrénujeme neuronovou síť, provedeme experimenty, které vyhodnotíme, a nakonec navrhneme použití pro budoucí aplikace a vylepšení.

Keywords

Speech synthesis, whispered speech, WORLD, BLSTM, conversion.

Klíčová slova

Syntéza řeči, šepot, WORLD, BLSTM, konverze.

Reference

GAJDA, Richard. *Conversion of Whispered to Normal Voice.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jan Brukner

Rozšířený abstrakt

Cílem je převést šepot na normální (znělým hlasem mluvenou) řeč za pomocí neuronových sítí a použití této konverze v mobilní aplikaci. Použitý postup je popsán ve vědeckém článku z Indického Institutu Věd v Bengalúru, který slouží jako předloha pro řešení popsané v této práci.

Tento postup má využití například při tísňovém hovoru, kdy volající v nouzi nemůže z jakéhokoliv důvodu mluvit nahlas, nebo pro lidi s anatomickou (ať už vrozenou, nebo externě zapříčiněnou) vadou hlasového ústrojí, nebo infekční chorobou napadající hlasivky a dýchací ústrojí.

Prvním úkolem je získání vhodného datasetu, na kterém se budou neuronové sítě trénovat. Tento dataset je vytvořen šesti řečníky, třemi ženami a třemi muži, kteří namluví zdrojový text jak šeptem, tak i znělou řečí.

Práce dále definuje techniky analýzy hlasu, čili získání hlasových excitačních a modifikačních příznaků. K této analýze je použit WORLD, systém pro analýzu a syntézu řeči založený na principu vokodérů. Následuje popis standartních postupů pro získávání příznaků a následně popis jednotlivých algoritmů, které používá WORLD. Příznaky jsou získávány jak ze znělé, tak I šeptané řeči. Jelikož je cílem predikovat znělé příznaky šeptané řeči, je tato šeptaná řeč upravována metodami analyzujícími spektrální chování a následně zarovnána pomocí dynamického borcení času.

Samotné konverzi předchází definování vrstev a aktivačních funkcí neuronových sítí, které se používají pro konverzi získaných spektrálních charakteristik šeptané řeči na jednotlivé příznaky řeči znělé.

Dále je rozebrán postup definovaný ve zdrojovém indickém článku a jeho nutné modifikace pro splnění cílů této práce. Zdrojový článek se nese spíše v experimentálním duchu, zatímco tato práce využívá výsledků těchto experimentů a aplikuje je do praxe.

V další kapitole je popsána implementace získávání a modifikace příznaků, trénování neuronové sítě a provádění inference na natrénovaých modelech. Následně jsou získané data z natrénovaných modelů porovnávány s originální znělou řečí. Výsledky naznačují, že modely fungují velmi dobře na zdrojových řečnících (řečnících z datasetu) a to jak za použití modelu trénovaného přímo na daném mluvčím, tak I za za použití modelu trénovaného na více mluvčích. Nicméně konvertovaný vstup od mluvčího, který není součástí datasetu, nabízí pouze omezené výsledky, kterým často nejde ani rozumět.

Následuje popis implementace cílové aplikace pro Android, která našeptanou zdrojovou řeč od uživatele odesílá na REST API server, kde je nad ní provedena analýza příznaků a konverze skrz natrénované modely. Tato konvertovaná řeč je zpětně poslána uživateli, který si ji může přehrát.

Conversion of Whispered to Normal Voice

Declaration

I hereby declare that this Bachelor's thesis has been written as an original work by the author himself under the supervision of Ing. Jan Brukner. I have listed all the literary sources, publications and other sources which were used during this thesis preparation .

.....
Richard Gajda
May 18, 2021

Acknowledgements

Special thanks go to Václav Dreiseitl from BOMB JACK Studio in Hranice na Moravě for his being so kind to assist me me with acquisition of used dataset. I would also like to thank my supervisor Ing. Jan Brukner for his expert insight into the subject and assisting me with any problem I have come upon.

Contents

Chapter 1

Introduction

The goal is to implement a procedure which processes whispered speech input and converts it into synthesized normal speech using neural networks. This procedure is described in the article published by G. Nisha Meenakshi from Indian Institute of Science, Bangalore [7], which will be used as a reference and guide. The objective of this thesis is to a develop an application, which will apply the implemented method on user request. Future uses for this application may be numerous: e.g. a user who recieves a whispered voice message on any messaging app, would like to convert the recording and play it back in normal voiced form. Another model situation for its usage may be an emergency situation distress call, when the calling person is in hiding and has to speak as quiet as possible. Yet another possible use is for a user with such a health condition that would not allow him/her to speak normally. This could be a condition ranging from sore throat to voice exhaustion, and, in extreme cases, if one's vocal chords are damaged or even severed.

Therefore, the goal of this thesis is to obtain a dataset for neural network training, to implement a conversion solution based on the reference article, and to develop a *mobile application* that is capable of processing the said conversion in a reasonable amount of time. This solution works with several technological methods and approaches for speech analysis, speech parameterization, neural network principles and mobile application development which are described in a theoretical chapter. The implementation chapter describes the architecture and addressing problems of algorithmization of theoretical practices, and implementation of user interface and functional components of the app. Lastly, fully functional models will be deployed into the application for usage, user evaluation and feedback.

Chapter 2

Theory

The solution of the problem required understanding many signal and speech processing concepts and methods. Understanding some of them proved to be absolutely critical for the application to work, others were not so crucial and required only a „need-to-know“ knowledge base. Such concepts are mentioned only briefly, because they are not part of the implementation (e.g. signal normalization, frame segmentation, feature extraction [19]), they are extracted from working libraries and frameworks (e.g. WORLD). The methods that are implemented are described in detail.

2.1 Voice Parameterization

Suppose the examined signal is already preprocessed and segmented into frames, parameterization (or „reading“ information) can begin now. It is important to know which information is important – in this case, it is desirable to know both excitation and modification parameters (or signal excitation parameters and frequency modulation). Although several conventional algorithms for feature extraction are mentioned for comparison purposes, WORLD vocoder-based native feature extraction algorithms are used exclusively in the proposed solution.

2.1.1 Fundamental frequency

Fundamental frequency, *F0* or *pitch* is the main foundation of the thesis. It describes the frequency of oscillation of vocal chords. This is an important speech characteristic, because whispered speech is devoid of it, whereas neutral speech is defined by it. F0 can be calculated using many varying methods and algorithms using temporal (e.g. autocorrelation) or spectral (Cepstrum) signal characteristics [13]. For this thesis purpose, the WORLD's *Harvest* [11] F0 estimation algorithm will be used.

Unlike conventional pitch obtaining auto-correlation function (ACF), defined by equation [19]

$$R(m) = \sum_{n=m}^{N-1} s(n)s(n-m) \tag{2.1}$$

which measures statistical intensity of relationship between two instances (in this case frames of segmented speech), Harvest algorithm proposes a different approach. Firstly, numerous band-pass filters, each with different center frequencies are placed on the input

waveform. Since F0 is unknown, many filters are required. Only the filters that output sine-shaped waves, are considered as basic fundamental frequency candidates. F0 candidates are then estimated from F0 basic candidates if they all output the same sine wave in a certain bandwidth. Such F0 candidates (many are overlapping) are further refined (using instantaneous frequency), the unwanted ones (F0 contour does not change rapidly in a fundamental period, therefore the F0 candidates with rapid changes above a set threshold are considered unwanted) are removed and then F0 contour is smoothed and returned.[11]

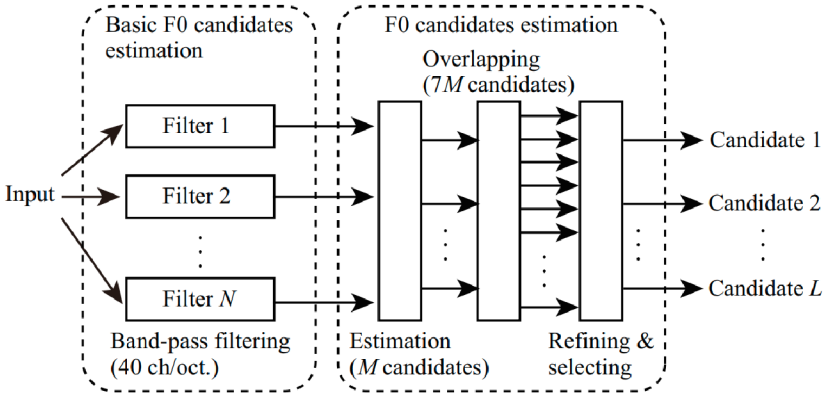

Figure 2.1: Harvest algorithm outline [11]

WORLD also allows further F0 contour refinement, using the WORLD's StoneMask algorithm. The StoneMask method is used to improve the noise robustness of the estimated F0 contour.

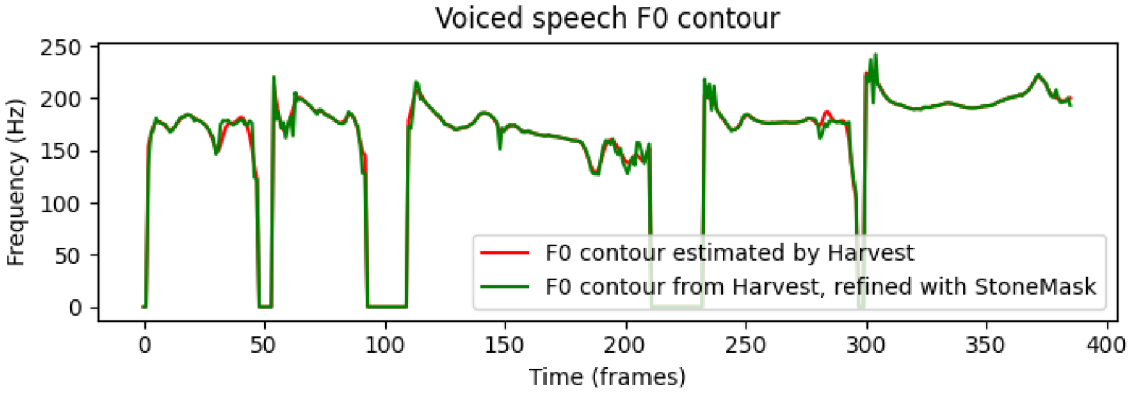

Figure 2.2: F0 contour estimated using Harvest, refined with Stonemask.

Since pitch is gender-dependent, the F0 values span roughly in the interval of: $90\ Hz < F0 < 120\ Hz$ for **male speakers** and in the interval of: $150\ Hz < F0 < 300\ Hz$ [19] for **female speakers**.

2.1.2 Aperiodicity levels (ap)

Since natural human speech does not contain only the periodic signal, produced by periodically vibrating vocal chords, another parameter is used to describe these aperiodicities. Aperiodic signal sources can be e.g. aspiration between the vocal chords, friction and transient discharge generated by loosening a contraction. The parameter has an important utilization in the field of speaker recognition systems and can drastically increase the quality of speech synthesis systems, however it is not used by default very much. The parameter can be obtained by several passive procedures (zero-crossing rate, high-low frequency ratios [3]), but since the WORLD vocoder with build-in algorithms (specifically D4C algorithm) has been used, all which is needed to know is the following:

"D4C uses a group-delay-based parameter. This parameter forms a sine wave of F0 Hz from arbitrary periodic signals with a fundamental period of T0. Therefore, the power ratio between the sine wave and the other frequency components corresponds to the aperiodicity." [10]

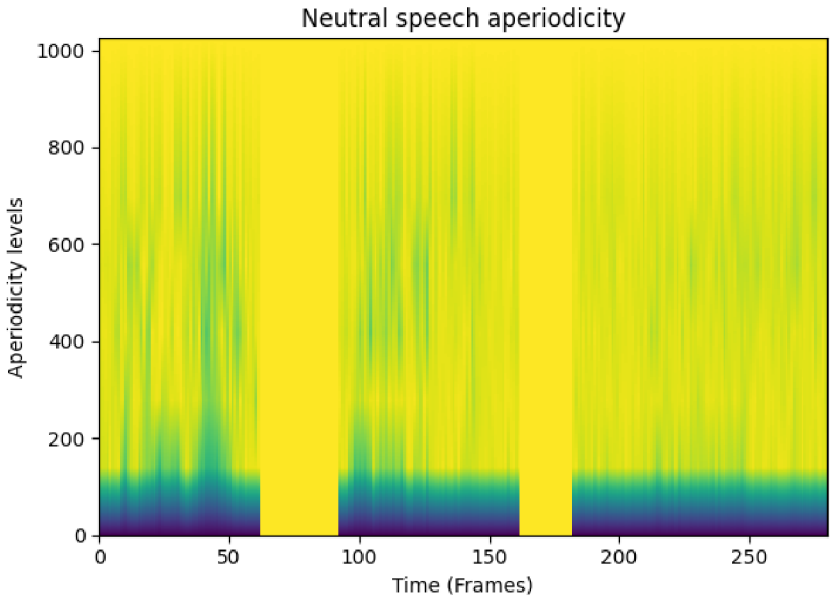

Figure 2.3: Aperiodicity levels of a neutral speech utterance obtained by D4C [10]

2.1.3 Spectral Envelope

Spectral Envelope (sp) is an encapsulating curve of the amplitude spectrum. To be precise, it gives the value at one certain point in time (in this case one window). Again, many algorithms (e.g. Cepstrum and linear predictive coding – LPC) can be used for estimation, but since those are heavily reliant on time variance, WORLD vocoder's own algorithm – *CheapTrick* [9] – will be used.

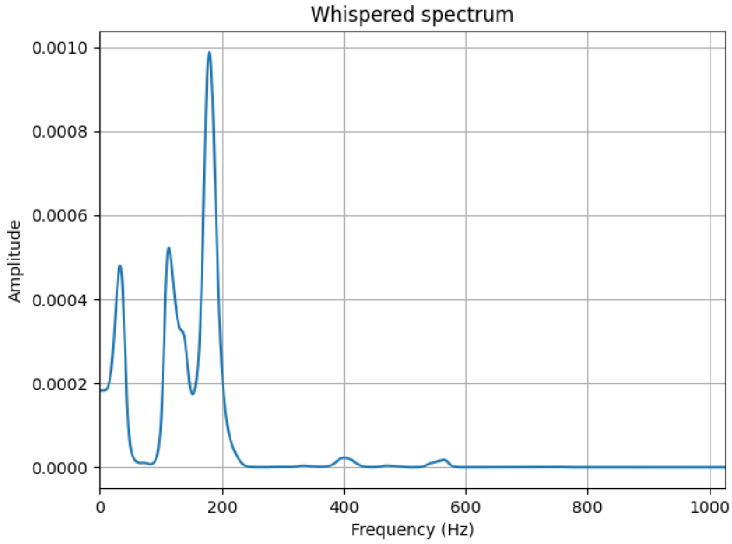

Figure 2.4: Spectrum of a frame of a whispered utterance

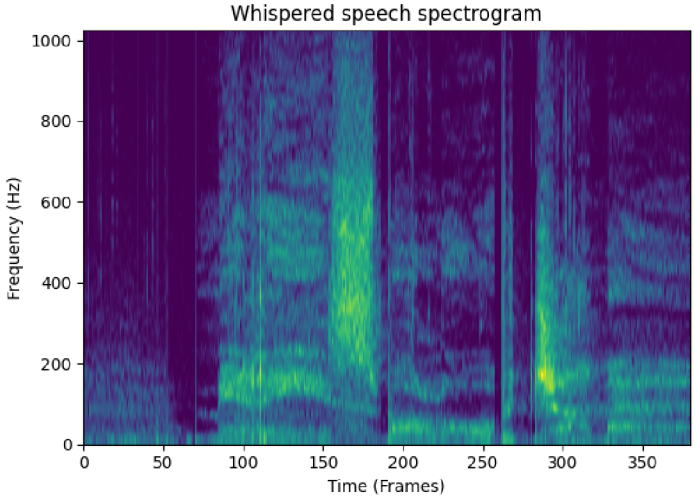

Figure 2.5: Spectrogram of the same whispered utterance obtained with CheapTrick [9]

CheapTrick design is based on conventional algorithms, namely F0 adaptive windowing and various cepstral methods. It consists of three components: F0-adaptive windowing, power spectrum smoothing and liftering processing for spectral reconstruction and smoothing. First, a windowing function is needed, that one which is ideal for the pitch analysis. A

Hanning windowing function is used for this method. Once the overall power of the periodic signal within the window is temporally stable, the power spectrum obtained from the first step can be smoothed. The power spectrum is reshaped into a logarithmic form to be used for working in the quefrency domain. Lastly, a liftering in the quefrency domain is done to remove the frequency variation, which is caused by discretization. Spectral recovery is done at the same time [9].

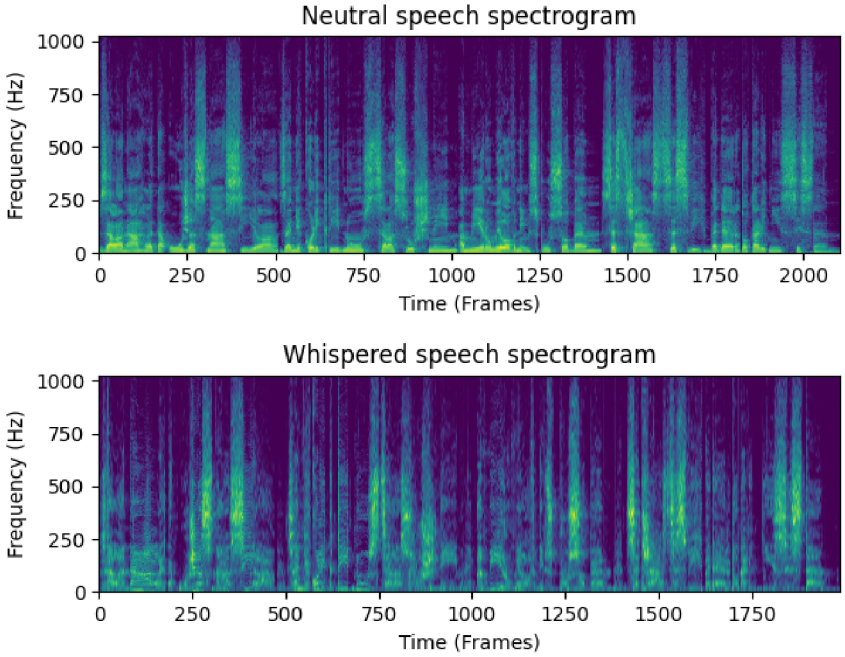

Figure 2.6: Neutral and whispered utterance spectrograms obtained using CheapTrick

As it can be seen in the bottom of both spectrograms, 2.6 there is a difference between neutral and whispered speech spectrum – a fundamental frequency (F0) is missing in the whispered spectrum.

Extracting spectral envelope for each frame is essential, since prediction of previous parameters based on the **sp** will be performed. Since the feature vector for each window contains too much information, MFCC calculation over the spectral envelope will be computed.

2.1.4 MFCC

The main issue that comes with the usage of cepstrums is that it uses DFT, which has the same resolution for all frequencies. That is undesirable in the terms of human hearing, which is not linearly sensitive to all frequencies. In the boundaries of speech analysis/synthesis it is the aim to be able to align cepstrum to hearing.

This is achievable by employing **Mel-Frequency Cepstral Coefficients** or MFCC, which will place non-linear filters along the frequency axis, measure energy at filter output and use that energy to calculate cepstrum, instead of using DFT. The used non-linear filter modification uses Hertz (F_{Hz}) to Mel (F_{Mel}) conversion which can be calculated with

following formula [19]:

$$F_{Mel} = 2959\log_{10}(1 + \frac{F_{Hz}}{700}) \tag{2.2}$$

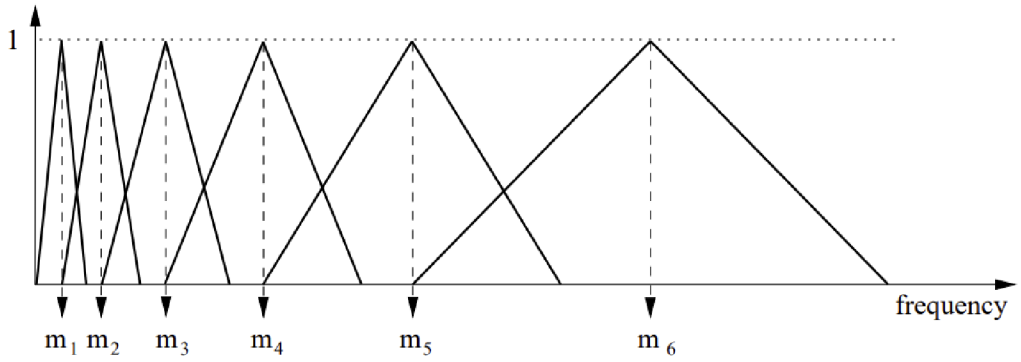

Figure 2.7: Placement of Mel filters on frequency axis [19]

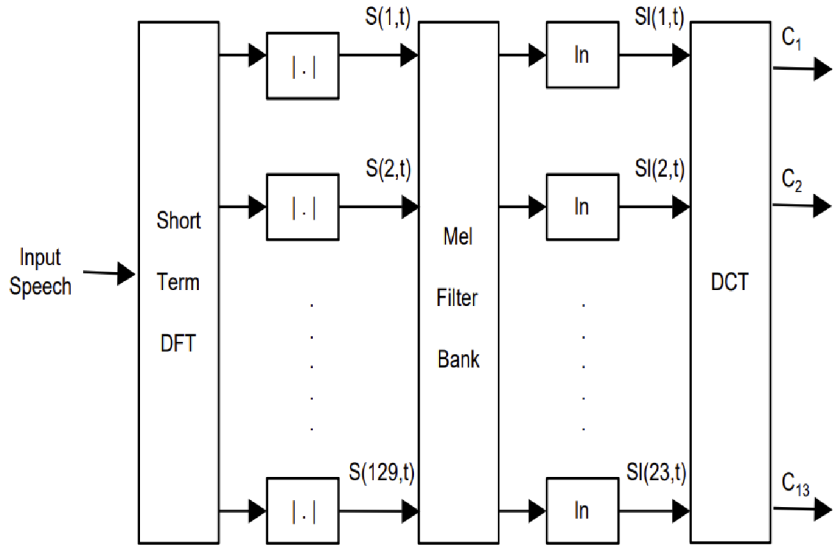

Figure 2.8: MFCC Computation - Block diagram [19]

This is a baseline method for spectral envelope coding. An advanced modification of Mel-generalized spectral analysis, native to WORLD vocoder, will be used. Spectral envelope obtained from CheapTrick algorithm is coded using WORLD coding algorithm. It is possible to use various frequency warping functions (e.g. Bark or ERB scale [9]), but for the purposes of this thesis, Mel scale is used. Used scale differs from the formula mentioned above only in the coefficients used. WORLD vocoder frequency warping formula for spectral envelope coding is as follows [12]:

$$F_{Mel} = 1127.01048 \log_{10}(1 + \frac{F_{Hz}}{700}) \tag{2.3}$$

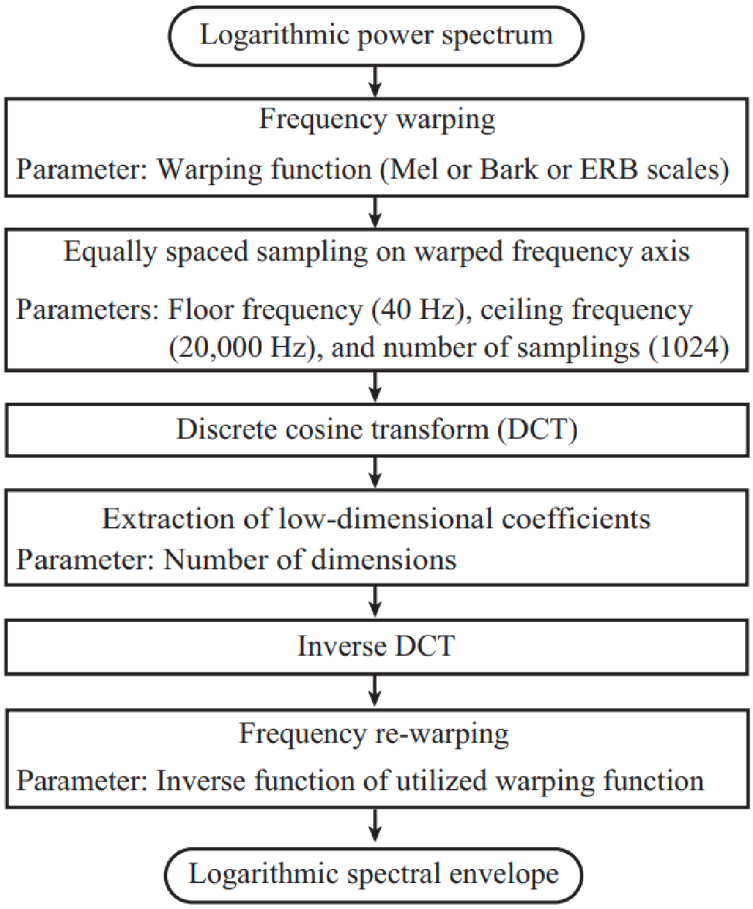

Figure 2.9: Outline of WORLD spectral coding [12]

Resulting features are called **MCEPs – Mel cepstrum envelopes**, which are the input of the Neural Network training system (each output feature prediction will be trained on whispered speech MCEPs). Noticable fact is that the only thing that is directly derivable from whispered speech MCEP is neutral speech MCEP. For predicting **F0** and **ap** additional information of the dynamic transitions of the signal between each window (based on the source article [7]) is needed.

Delta Features

Such dynamic information can be obtained from computing Δ and $\Delta\Delta$ coefficients. The idea is to compute the difference (Δ) of each frame of the feature. The interpretation of such features is that they approximate the first and second derivatives of the signal [1]. The computation of deltas for the feature vector f_k and the time instance k is the following:

$$\Delta_k = f_k - f_{k-1} \tag{2.4}$$

$$\Delta\Delta_k = \Delta_k - \Delta_{k-1} \tag{2.5}$$

Delta feature vectors computed for each frame of MCEPs are then appended to MCEP vector (for each frame). That gives enough information about the whispered speech dynamics as well as the spectral changes. However, it is nearly impossible for the speaker to record the whispered and neutral utterances of identical length which is required for effective neural network training. To solve this issue, a DTW method is applied.

2.2 Dynamic Time Warping – DTW

Both whispered and neutral paired utterances need to be time aligned prior to parsing to the neural network. This can be done using a DTW algorithm. The principle of this method is that an existing dictionary of reference matrices for words, that need to be aligned, is given. Then a testing matrix of parameters (word to be aligned) is parsed to the aligner. Resultingly, it should be possible to say which of the reference words matches the tested word. This would be much easier, if words could be described using one vector only. But as it has already been mentioned, multiple vectors of vectors – matrices are being processed. DTW allows to adjust the vectors nonlinearly by employing a nonlinear time warping function. In practice, all elements of the tested utterance with given element of reference utterance and its neighbours are compared.

If a general time variable k is given, two transformational functions can be created:

- $r(k)$ for reference sequence
- $t(k)$ for testing sequence

Thus, the resulting **warping path** describes the two vector alignment. From such a path, the $r(k)$ and $t(k)$ function courses are derivable, and the sequences can thus be described step by step [19].

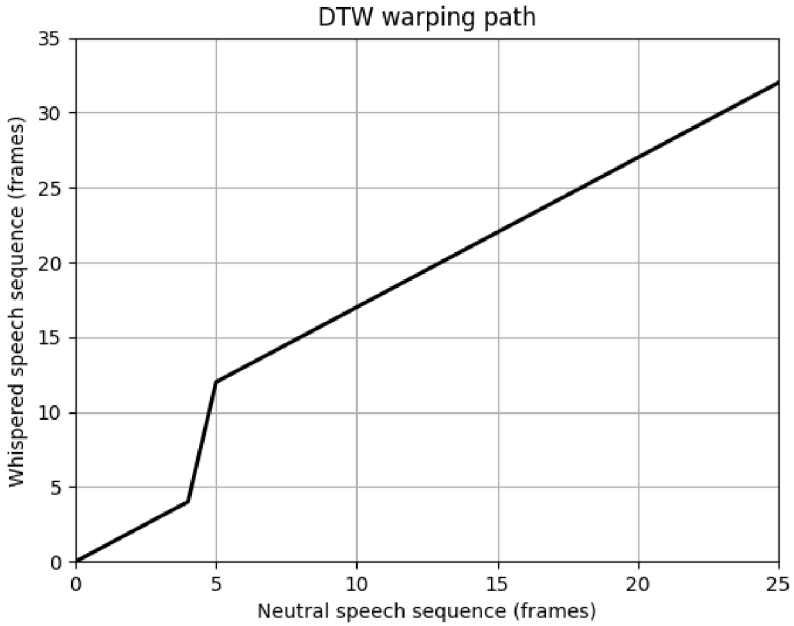

Figure 2.10: Path for aligning two sequences

Given two sequences X and Y of lengths $N \in \mathbb{N}$ and $M \in \mathbb{N}$, an (N, M)-warping path of length $L \in \mathbb{N}$ is a sequence $P = (p_1, \ldots, p_L)$ with each element of P satisfying boundary, monotonicity and step-size conditions. Boundary condition guarantees that the first and last elements of X and Y are aligned to each other. The monotonicity enforces timing: if one element of X is followed by another element of X, it should be also applied to their corresponding Y elements. Ultimately, step-size condition satisfies the continuity requirement. No element can be skipped, and there can be no recurrence in alignment.

Next, the warping path quality should be measured. To achieve that, numerical comparison of the elements of the sequences has been done. $\mathcal{F}$ is a feature space and assuming $x_n, y_m \in \mathcal{F}$, comparison of those two features is possible if a local cost measure exists:

$$c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}. \tag{2.6}$$

The cost is usually low, if two compared sequences are similar one another, which is the case, in this thesis boundaries. To evaluate the cost measure, a cost matrix $C \in \mathbb{R}^{N \times M}$ is obtained and defined as follows

$$C(n, m) := c(x_n, y_m) \tag{2.7}$$

The total cost $c_P(X, Y)$ of a warping path P related to the local cost measure c is calculated using following formula

$$c_P := \sum_{\ell=1}^{L} c(x_{n_\ell}, y_{m_\ell}) = \sum_{\ell=1}^{L} C(n_\ell, m_\ell). \tag{2.8}$$

Now, the aim is to obtain an optimal warping path between X and Y, which is such a warping path that has the lowest overall cost in the boundaries of all viable warping paths[14].

The DTW algorithm is described in the following figure:

Algorithm: DTW

Input: Cost matrix $\mathbf{C}$ of size $N \times M$
Output: Accumulated cost matrix $\mathbf{D}$
Optimal warping path P^*

Procedure: Initialize $(N \times M)$ matrix $\mathbf{D}$ by $\mathbf{D}(n,1) = \sum_{k=1}^{n} \mathbf{C}(k,1)$ for $n \in [1:N]$ and $\mathbf{D}(1,m) = \sum_{k=1}^{m} \mathbf{C}(1,k)$ for $m \in [1:M]$. Then compute in a nested loop for $n = 2,\ldots,N$ and $m = 2,\ldots,M$:

$$\mathbf{D}(n,m) = \mathbf{C}(n,m) + \min\{\mathbf{D}(n-1,m-1), \mathbf{D}(n-1,m), \mathbf{D}(n,m-1)\}.$$

Set $\ell = 1$ and $q_\ell = (N,M)$. Then repeat the following steps until $q_\ell = (1,1)$:

Increase ℓ by one and let $(n,m) = q_{\ell-1}$.
If $n = 1$, then $q_\ell = (1,m-1)$,
else if $m = 1$, then $q_\ell = (n-1,m)$,
else $q_\ell = \text{argmin}\{\mathbf{D}(n-1,m-1), \mathbf{D}(n-1,m), \mathbf{D}(n,m-1)\}$.
(If 'argmin' is not unique, take lexicographically smallest cell.)

Set $L = \ell$ and return $P^* = (q_L, q_{L-1}, \ldots, q_1)$ as well as $\mathbf{D}$.

Figure 2.11: Used DTW algorithm defined in *Fundamentals of Music Processing* [14]

2.3 WORLD Vocoder

For this thesis purposes, the vocoder-based high-quality speech analysis and synthesis system, called WORLD, will be used. The reason for this decision is that it is capable of near real-time speech processing and synthesis, which is crucial for the application purposes. The experiments made by its developers have shown irrefutable superiority over any other existing systems (e.g. STRAIGHT [6]) in terms of both speech quality and processing time [13].

2.3.1 Signal analysis

The WORLD vocoder is capable of using various feature extraction algorithms, which can be generalized into three different categories: Fundamental frequency (F0) estimation algorithms, spectral envelope (sp) estimation algorithms (which generally does not use only the waveform but also the F0 obtained in advance), and aperiodicity extraction algorithm (which computes aperiodicity directly from the waveform, F0 and sometimes sp).

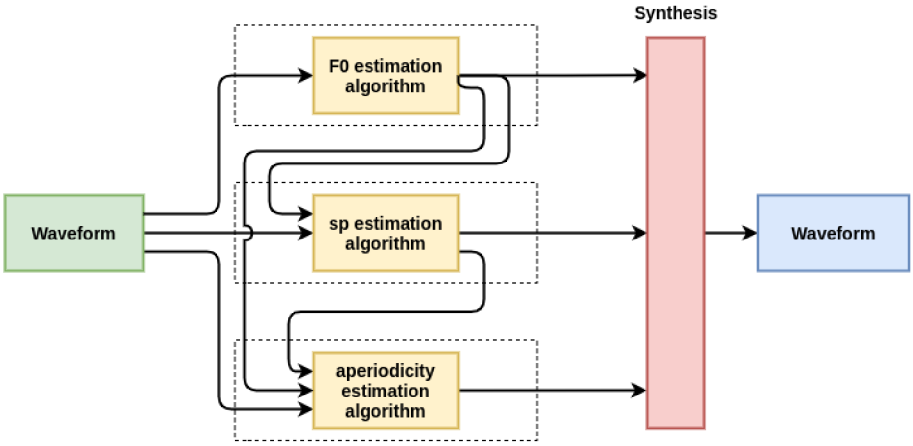

Figure 2.12: WORLD vocoder-based system overview [13]

2.4 Neural Network

As the core part of proposed application, it is required to have the obtained whispered speech features converted into those of neutral speech, so that the relations between them can be observed and described. Such relations need to be mathematically expressible. To achieve this, recurrent neural networks (RNN) – specifically **BLSTM – Bidirectional Long Short Term Memory** in combination with Linear Transform layers are employed.

2.4.1 Activation functions

LSTM layers use different activation functions. One of them is sigmoid function. The function applies this element-wise operation:

$$\sigma(x) = \frac{1}{1 + exp(-x)} \tag{2.9}$$

The second activation function is tanh function as an input shaper:

$$tanh(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \tag{2.10}$$

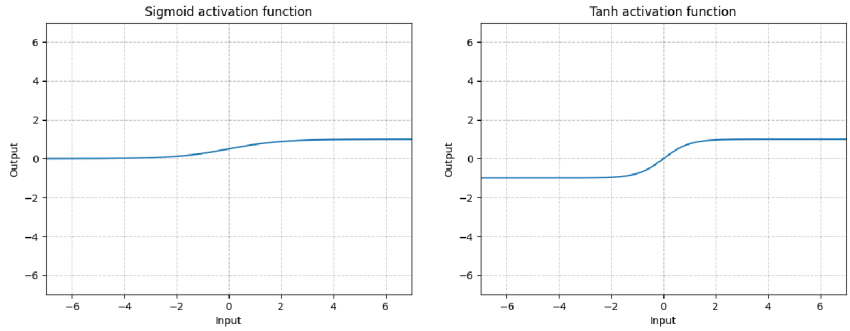

Figure 2.13: Sigmoid and tanh activation functions [15]

2.4.2 Layers

- *Linear Transform layers* are the most straightforward and easy to use layer types. The layers work as a multiplier of input data with weights and their adder. The linear transformation on the input data is applied as follows [15]:

$$y = xA^T + b \tag{2.11}$$

Where x is the input sequence, y is the output sequence. A^T and b are weights to be trained.

- *LSTM layers'* most notable feature is that they allow to remember the previous states of the NN, so each neuron (cell) has two inputs – one input from the data, and the other describing previous cell state. The network is thus able to work with a longer context of time. BLSTM is a modification of an LSTM layer that is not only connected to the past, but also to the future. The basic idea of BLSTM is to pre-send each training sequence forwards and backwards into a pair of separate RNNs which are connected to one unique output layer. That way, the network has full information about all points before and after time t at any given point in t of the sequence. [18]

Since PyTorch [15] is being used for the purposes of this thesis, a following computation is performed on every element of the input sequence:

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t),
\end{aligned} \tag{2.12}$$

"where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, h_{t-1} is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function and $\odot$ is the Hadamard [17] product [15]."

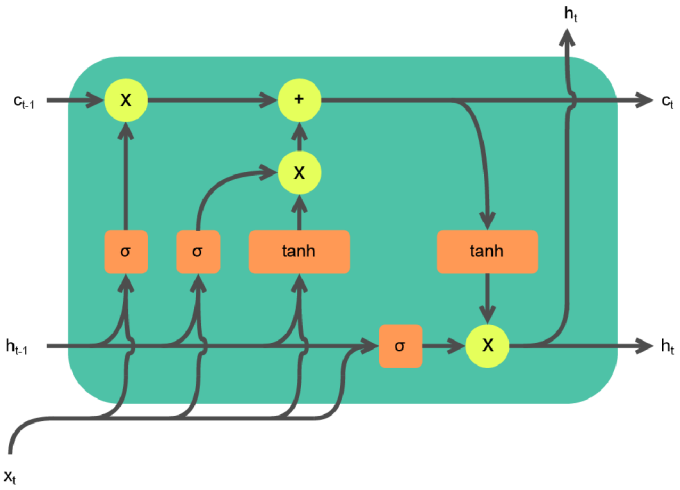

Figure 2.14: LSTM Schematic [2]

Chapter 3

Conversion of whispered speech to neutral

3.1 Existing method description

This thesis builds on the work of an existing paper [7]. The procedure described in the article uses BLSTM neural networks in order to predict behaviour of synthesized voiced parameters. The original method used the **STRAIGHT vocoder** [6] as a tool for extracting features and later synthesis. The idea was to obtain spectral envelopes of both whispered and neutral speech, converting them to MCEPs and computing the Δ and $\Delta\Delta$ features for whispered utterances. In the next step, using a DTW obtained path, features from both whispered and neutral utterances are aligned, so both are of exactly the same length. Then all aligned whispered features are taken and passed into separate BLSTMs – each for one of neutral features – namely aperiodicity, spectral envelope and base frequency. The method also calculates with a *vuv* parameter, which describes voiced – unvoiced phoneme decisions. The BLSTMs are then trained to predict neutral speech parameters based on fed whispered data. When the training is completed, the models can be used to convert whispered parameters into voiced ones, which are then used to synthesize a neutral utterance through STRAIGHT vocoder.

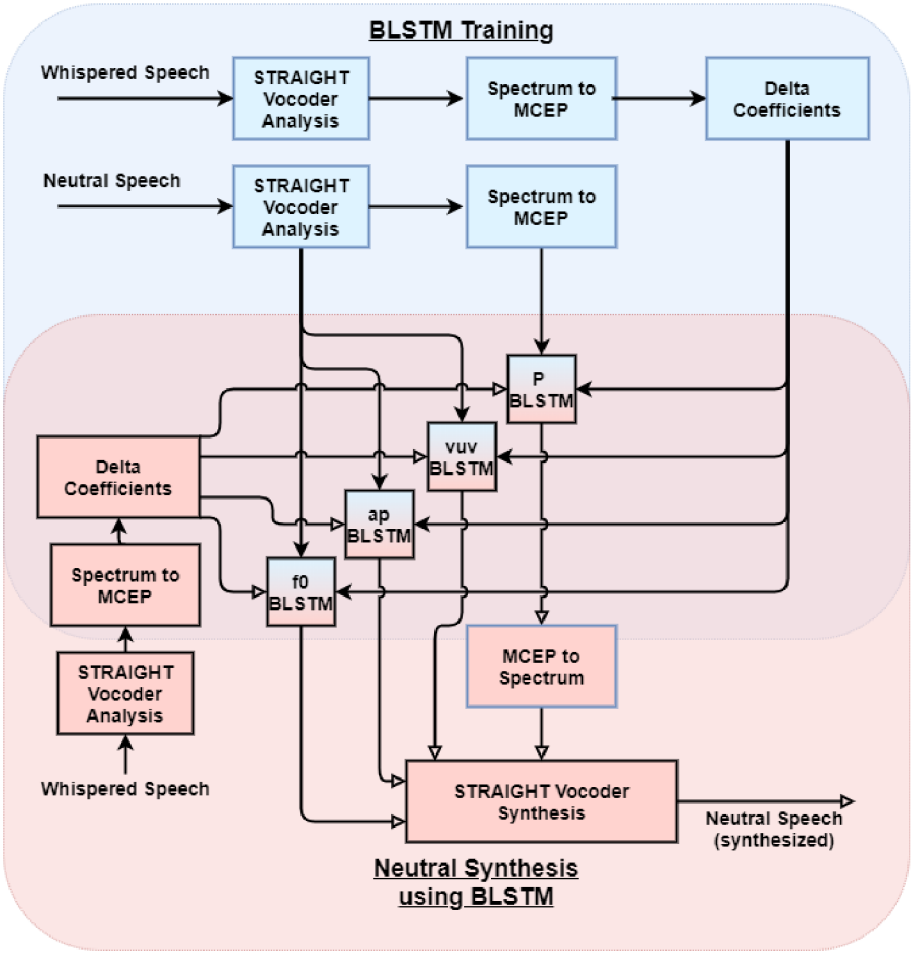

Figure 3.1: Illustration of existing method for training and employing BLSTMs

3.2 Modification of proposed method

The approach to certain aspects of solution is different than those presented above. The first distinction is a usage of different vocoder. Since near real-time processing is desirable, the STRAIGHT [6] vocoder, used in the model work, is not suitable. That is why the WORLD vocoder-based speech analysis system will be used. Since WORLD uses different algorithms for speech analysis, it returns the spectral envelope, aperiodicity levels and the fundamental frequency only. Therefore, instead of training four BLSTM models, only the three of them will have to be trained. Unlike in the experimental paper, the efficiency and the best results possible (provided by the paper) will be focused on. Therefore, one LSTM layer, 2 hidden layers with 256 cells and one linear output layer will be used for each model. The input layer for *F0* and *ap* BLSTMs is of dimension 78, consisting of 26 dimensions for MCEP of whispered speech and 26 + 26 for Δ + $\Delta\Delta$, respectively. Input layer for the MCEP training consists of 50 + 50 + 50 instead based on informal evaluation after the first training.

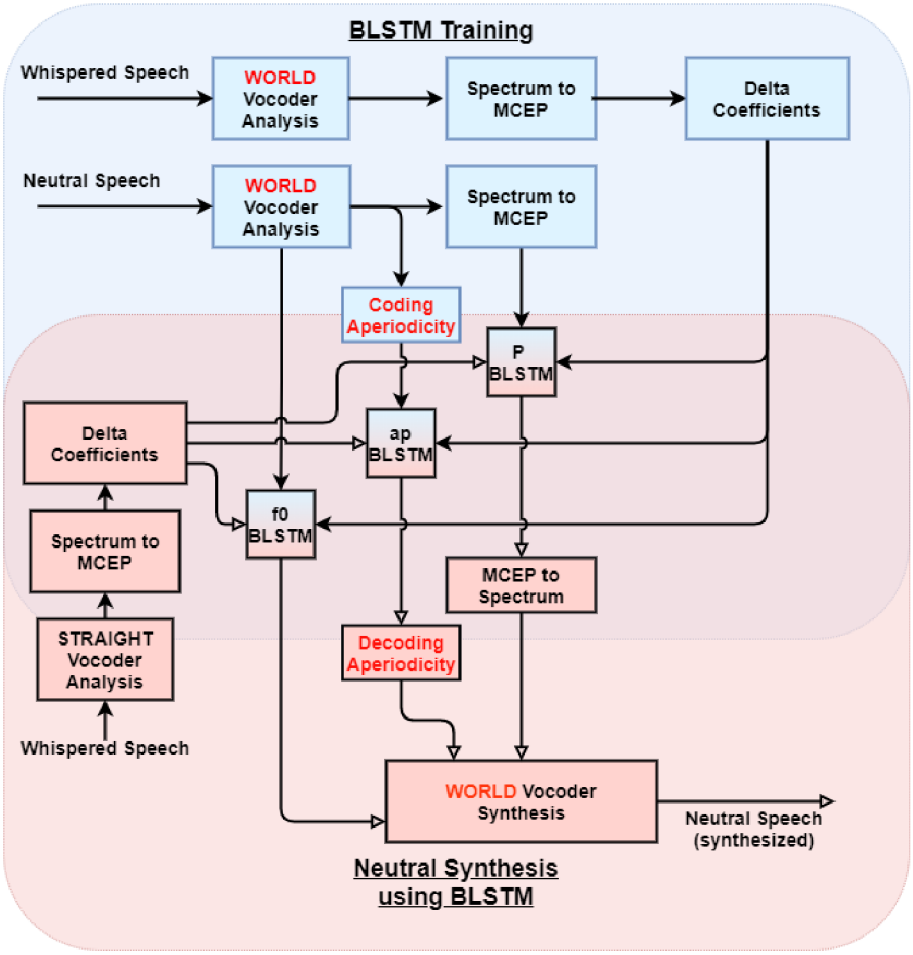

Figure 3.2: Illustration of modified procedure for training BLSTM and predicting neutral features using BLSTMs and synthesis (notice the missing vuv parameter)

3.3 Goals

Although the aim of the original paper was purely experimental and research based, this thesis' approach focuses on practical use of the proposed method. Specifically, the aim is to develop a smartphone app, with an intuitive and responsive user interface, for trained models evaluation and on-demand user-provided whispered utterances conversion.

Chapter 4

Dataset

The first challenge was to find a dataset suitable for the purposes of this thesis. Although the Indian Institute of Science in Bangalore were willing to help, they refused to provide the dataset, so putting together an unique dataset tailored for the purposes of this thesis seemed to be a better option (not to mention the fact that the dataset from the original article was recorded in a different language). The same setup was used – three male and three female speakers. The utterances were recorded by professional actors from National Theatre Brno. They all were asked to record a 1990 New-Year's speech by Václav Havel, the first democratic Czechoslovakian president after the Velvet Revolution, in 1989. The speech contains 2390 words contained in 121 sentences. The source text was recorded twice by each speaker – once speaking naturally and once whispering. Manual examination followed, consisting of editing and rarely discarding mispronounced utterances, removing excessive inhaling and exhaling and cutting into utterances of varying lenghts. It is important to note, that each speaker has developed a certain way of recitation, therefore number of utterances and their lenghts (after silence removal) vary for each speaker as listed below:

Utterance Table			
Speaker	No.utterances	Neutral duration	Whispered duration
Male 1	243	15m 32s	11m 49s
Male 2	241	13m 32s	11m 56s
Male 3	305	14m 8s	13m 9s
Female 1	312	15m 7s	13m 22s
Female 2	323	15m 47s	14m 58s
Female 3	322	18m 57s	18m 41s

Table 4.1: Number of utterance pairs and their respective lengths

The recording took place in a acoustically treated rooms in a recording studio. Following audio signal path was set up: Neumann U87i condenser microphone with a cardioid directional characteristic. Signal was then passed through a Warm Audio WA73-EQ preamp with no EQ in place and through a Warm Audio WA-2A compressor with slight compression on. Afterwards, the signal has been digitally processed using MOTU AD/DA converters and USB audio interface and recorded into Cubase 8 Pro DAW software. As for creating the source files, a 44.1 kHz sampling frequency and a 24-bit depth have been used.

Chapter 5

Implementation

5.1 Pre-processing

It is important to note, that many of the further mentioned procedures have been performed on encoded input waveforms, using PySoundFile library. The processed signals are NumPy [5] n-dimensional arrays (arrays of arrays). For simplification purposes, these multidimensional vectors will be further referenced only as „arrays“.

5.1.1 Silence Removal

All manually edited utterances have had their sequences of silence removed prior to WORLD analysis. First, a windowing function has been implemented. Then short-term energy [19] has been calculated for each frame

$$E = \frac{1}{l_{frame}} \sum_{n=0}^{l_{frame}-1} x^2[n] \tag{5.1}$$

where E is the short-term energy, l is the number of frames in the utterance, n is the current frame and x is the amplitude of the current frame.

Short term energy has then been compared to a threshold constant (which has been set to 0.0025). The input array elements, that are marked as noise have been removed and a waveform that is rid of silence has been returned.

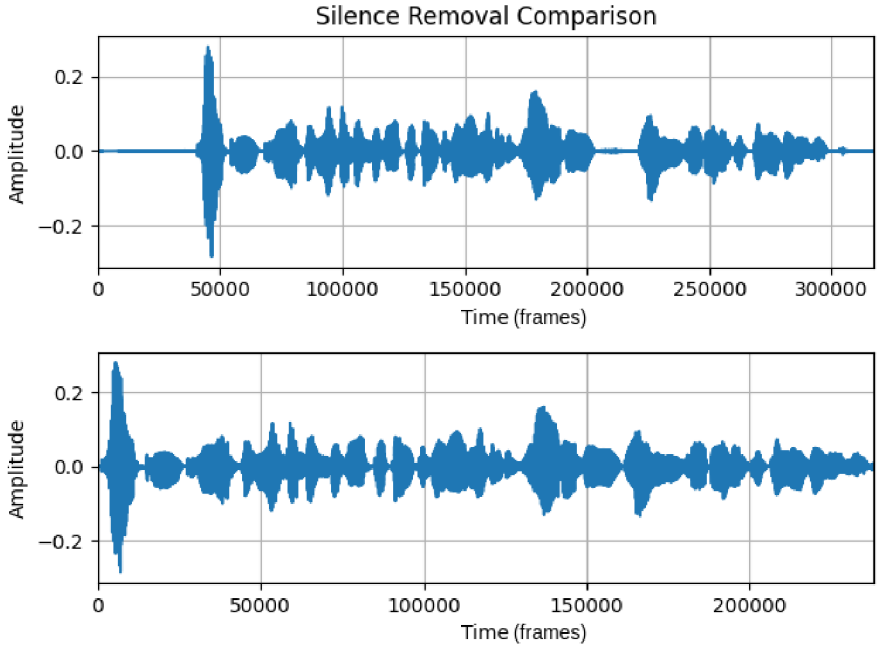

Figure 5.1: Silence removal comparison for neutral speech utterence from dataset.

5.1.2 WORLD Analysis and whispered speech preparation

As all the utterances pairs have been rid of silent parts, loading them into WORLD was the following step. PyWorld wrapper [1] has been used for easy array data manipulation. The waveform array from PySoundFile (x), along with the 44.1 kHz recording sampling frequency(fs) has been parsed into the PyWorld harvest method, which extracts fundamental frequency $F0$ and the time context (frames information). Frame overlap is set to 5 ms. This procedure is the same for both whispered and neutral speech utterances. From this moment forward, a different implementation will be done on neutral speech and different on whispered speech.

Neutral speech WORLD analysis

As the 2.12 outline suggests, the input waveform x_n, the sampling frequency fs_n the fundamental frequency $F0_{nh}$ and the time context t_{nh} (n standing for neutral speech, h for harvest method) are parsed into different methods to compute other speech parameters. The CheapTrick method is called to obtain spectral envelope, the D4C for aperiodicity feature and the StoneMask for refining the F0 curve. The coding of spectral envelope is called upon next, with input parameter m standing for target number of MCEPs, returning the desired MCEPs for BLSTM training. It is important to note that unlike in the source article [7], this method requires coding the aperiodicity parameter in a similar manner.

[1] https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder

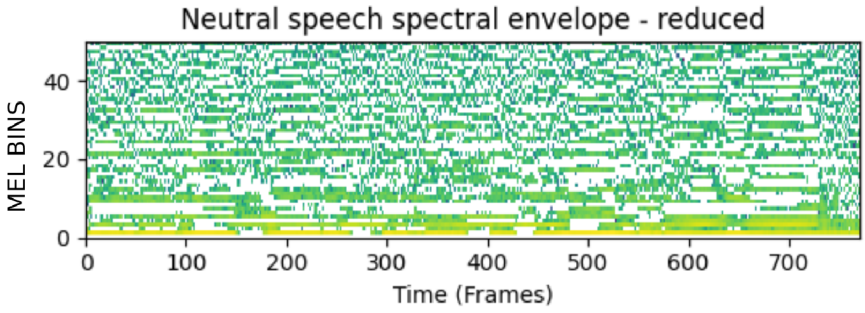

Figure 5.2: Spectral envelope of neutral utterance after coding [12]

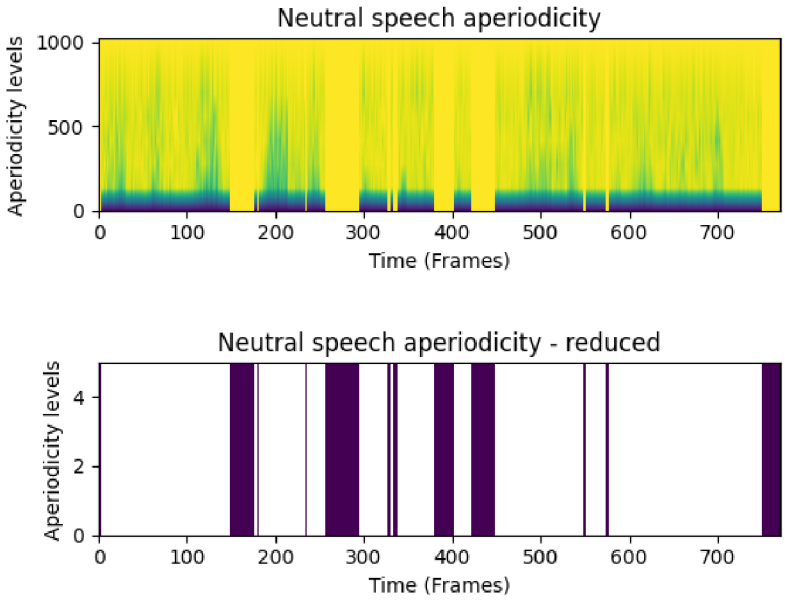

Figure 5.3: Aperiodicity of neutral utterance before and after coding

At this point, neutral utterances have been parameterized, outputing arrays with dimensions of 1, 5 and 50, for $F0$, ap and $MCEP$ respectively.

Whispered speech WORLD analysis and preparation

Similar to neutral speech parameter extraction, the four parameters x_w, fs_w, $F0_{wh}$ and t_{wh} are used to obtain a spectral envelope using CheapTrick. The spectral envelope is then encoded into MCEPs in the same fashion as in neutral speech. Δ and $\Delta\Delta$ features are computed from the MCEP array [2]. The delta function works with 2 frames before the current frame, and the 2 following current frame. After that, Δ and $\Delta\Delta$ values of all MCEP array indexes are concatenated with the corresponding MCEP array indexes. The resulting array shall be referred to as $w_{features}$ from this point forth.

[2] https://https://github.com/jameslyons/python_speech_features

The next step is to employ DTW to align the lengths of whispered speech to neutral speech using warping path. For that, a Matt Shannon's library is used [3]. First, a cost matrix is calculated from $MCEP_n$, $MCEP_w$ and cost function. An euclidean cost function is used.

$$eucCost(x, y) = \sqrt{(x - y)^2} \tag{5.2}$$

Subsequently, the cummulative cost matrix which stores all cost matrices, is calculated. From those cost matrices, minimal cost matrix is selected and best path is returned.

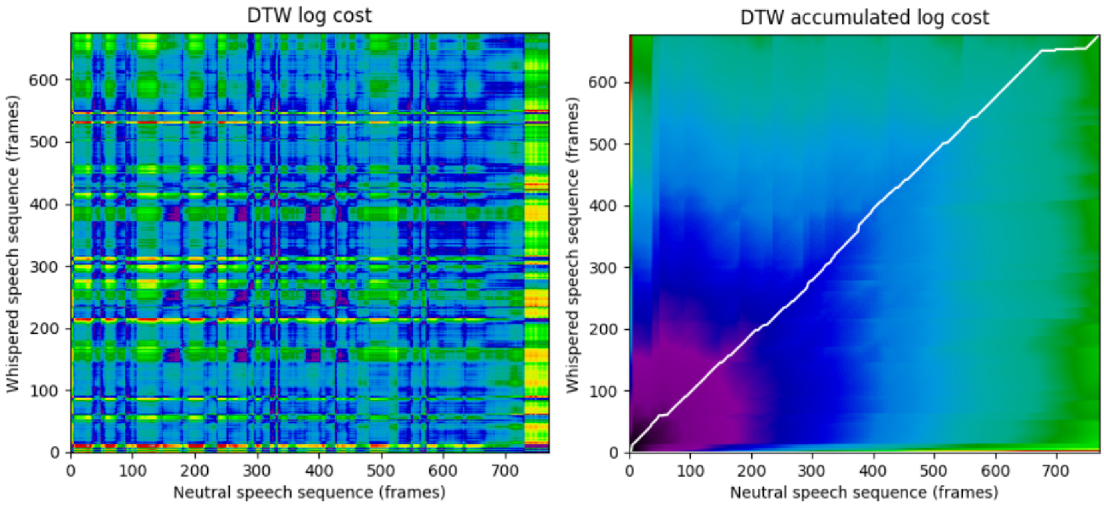

Figure 5.4: DTW log cost matrices, along with optimal warping path

Now that the $w_{features}$ sequences are aligned with neutral features, they are saved using the NumPy compressed save method. That is done because it is faster to access the data in the form of the NumPy arrays, rather than having to open the source waveform files all over again. Each file is named using xxx_{ds} format, where the xxx stand for utterance index number. Each utterance file contains $F0_n$, ap_n, $MCEP_n$ and $w_{features}$ arrays (which are all the same length). The saving operation concludes the preprocessing phase.

[3] https://https://https://github.com/MattShannon/mcd

5.1.3 Dataset building

Since the PyTorch deep learning framework is being used, the input data needs to be further processed into corresponding format. Two datasets types need to be constructed – an input ($w_{features}$) dataset and a target ($F0$ || ap || $MCEP$) datasets. Here, two options of feeding data into the LSTM architecture are presented. One option was to approach the input sequences frame-wise. That being said, LSTM would only work within a narrowed data context data and possibly would not be able to converge to a desired output, at all. For instance, in a shuffled dataset input, there is a possibility, that the network would recieve only one type of speech signal (e.g. periodic), therefore it might misinterpret next batch of data, which would consist only of aperiodic signals. To eliminate that, the second, utterance-wise approach is selected. Such approach (called mini-batch sequential training) should be drastically faster than using `batchsize=1`. To use mini-batch sequential training, all sequences must have the same lengthS. For that, a padding function is employed. Firstly, all the speaker's utterances are iterated in search of maximum length utterance. For example, the first male speaker's longest utterance consists of 2319 frames. Then a second iteration over the whole utterance set is made, extracting all the features, getting the utterance length, padding the features (adding zeros at the end of array up to `array[2318]` and saving both length and padded features into a file, once again using compression method. The resulting files are saved in xxx_{pad} format now.

The dataset constructor from PyTorch' `torch.utils.data` is used. It is important to note that the method does not keep all the data in memory, but rather loads features on demand during iterations in a neural network loop. Firstly, it requires speaker and mode (either *train* or *test*) specification. That specification is required only for the source files localization purposes. Next, the compressed padded data files are opened and the features extracted. At this point, it is important to mention that three different dataset constructor classes are implemented: `TrainF0Dataset, TrainAPDataset, TrainMCEPDataset`. Each constructor extracts $w_{features}$, $utterance_{length}$, $utterance_{index}$ and their respective features. Any needed data type or array shape conversion is done, so when the `getitem` method is called, all values are now returned as tensors.

5.2 Neural network

A PyTorch Module that defines the model is being implemented. This model consists of two layers: one LSTM layer that processes the inputs into hidden layer and one linear layer that processes the hidden layer into target outputs. The LSTM layer uses `hiddensize=256` with two hidden layers. Bidirectional mode and batch first are enabled and dropout of 0.1 is used. Mean Square Error (MSE) loss function is selected. The input dimension is set to $MCEP_{wsize} + \Delta_{size} + \Delta\Delta_{size}$. The linear layer takes the double of the hidden layer size (that's because bidirectional LSTM is used) as an input dimension, and varying output dimension, based upon target feature being trained. Forward method uses the torch `packpaddedsequence` for mini-batch sequential training. That is the reason to keep utterance lengths in our dataset. Those packed sequence tensors are parsed into the LSTM layer. The torch `padpackedsequence` is then used on LSTM output and that output is parsed into the linear layer.

Datasets for each neutral speech features are loaded using the PyTorch `DataLoader`. The sequences in batches are shuffled. Each dataset is divided into three parts: Train dataset, test dataset and validation dataset. Ten utterances are selected as test samples on which the neural networks performance will be tested. The remaining utterances are divided in a 9:1 ratio as train and validation dataset. That number (as opposed to the original article, which uses 3:1 ratio [7]) was chosen because the source datasets have not been very numerous, so the highest possible amount of train utterances is desirable. It is important to note, that both individual and (gender) generalized models have been trained – specifically, to demonstrate the difference between the conversion of out of dataset whispered speech into neutral speech using the one-speaker model only and the attempted generalized model.

5.2.1 Training loop

As per training loop, firstly a corresponding feature model has been initialized, then passed into CUDA environment. Datasets are loaded separately, for both training and testing phases. Then epoch-wise iteration was started. From experimenting with different training loop parameters and model layer sizes, 750 epochs were selected as a sufficient number to completely train target BLSTM models. Each epoch starts with sorting sequences in a batch length-wise. Hidden states are initialized and all variables are passed onto CUDA environment as well. As the forward function is called, the optimizer performs a parameter update based on the current gradient and the update rule. Model's state dictionary is being saved every 20 epochs, in case of overtraining the network. Running loss is being calculated for each phase (test and train phase) at the end of each epoch, so the record of loss history can be kept for selection purposes.

$$loss_{phase} = \frac{running\_loss}{dataset\_size_{phase}} \tag{5.3}$$

Its visualization allows selecting the optimal network model's state.

5.2.2 F0 models training and inference

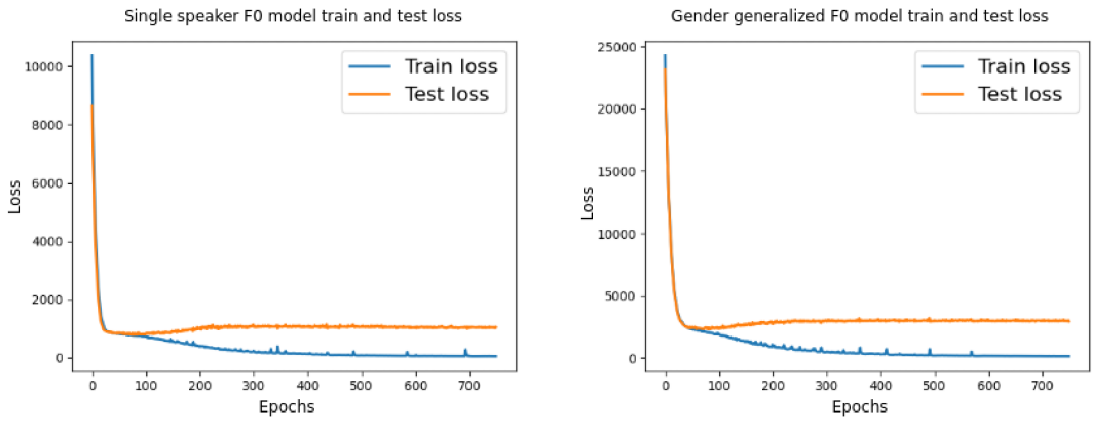

Figure 5.5: F0 loss differences between single and multiple speaker models

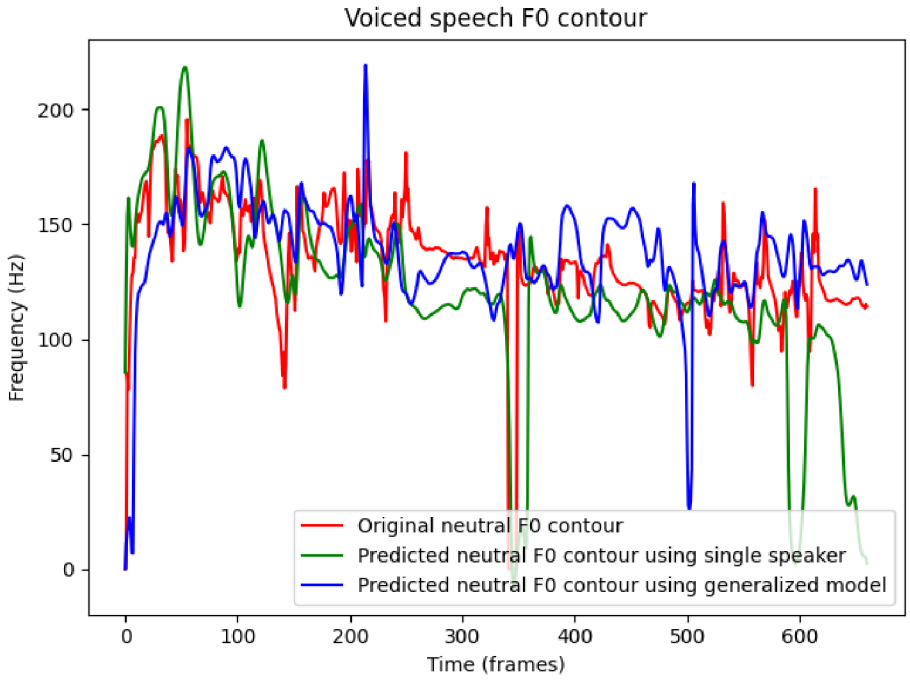

Figure 5.6: Inference comparison of F0 models

As expected, generalized models struggle with fundamental frequency prediction (the loss is significantly higher), because each speaker has a different pitch. Inferencing pitch using a single speaker model for the out-of-dataset whispered utterance results in a converted utterance resembling the dataset speaker. Whereas using the generalized model does result in a conversion that resembles quick changing through the pitch of the three dataset speakers, rather than the input out-of-dataset speaker's natural voice.

5.2.3 Aperiodicity models training and inference

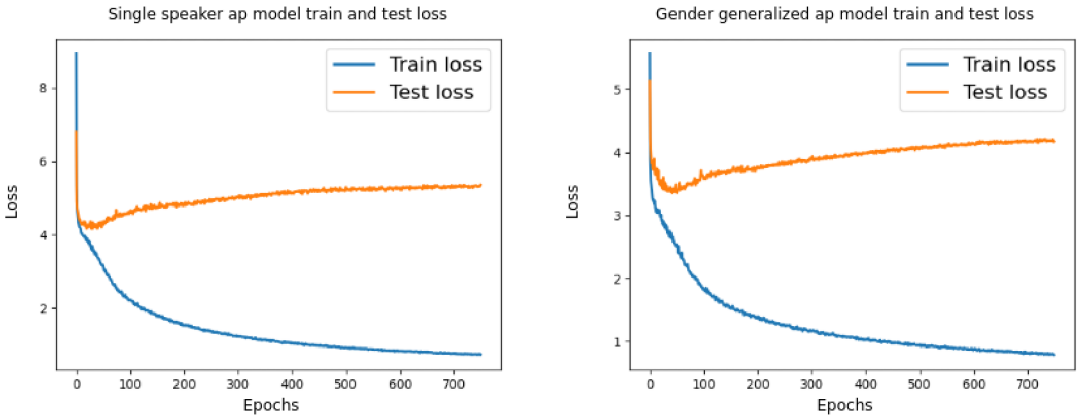

Figure 5.7: Aperiodicity loss differences between single and multiple speaker models

It should be stressed, that the dataset built for F0 training has not been normalized, resulting in longer training to achieve desired outputs. Aperiodicity training was executed on normalized data (values ranging from -1 to 1) and, therefore, as the graph suggests, better results in earlier epochs of training have been achieved. After restoring the full dimensionality, graphic interpretation has shown that aperiodicity prediction worked rather nicely, with generalized models showing more detailed results.

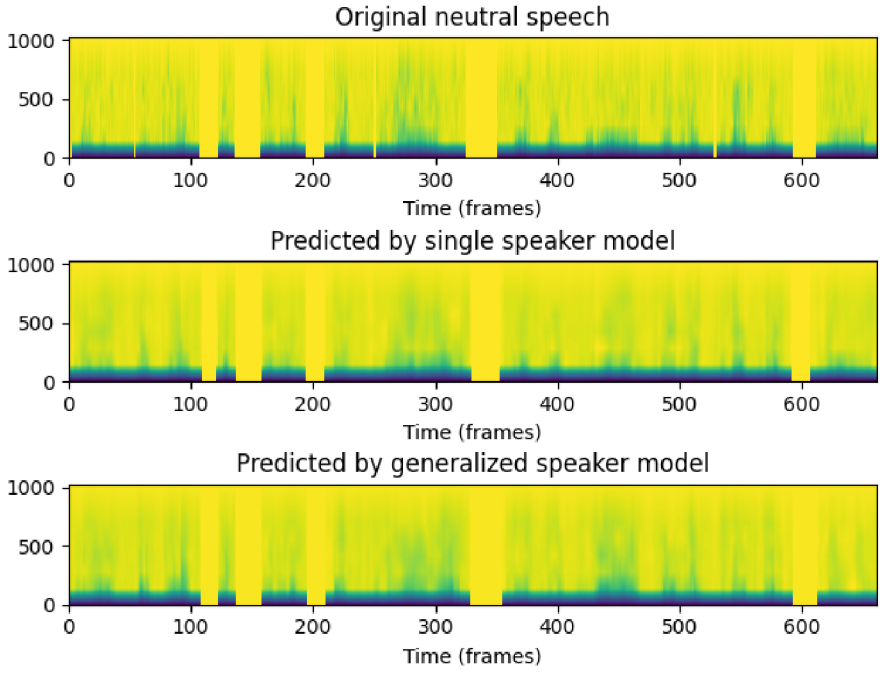

Figure 5.8: Inference comparison of aperiodicity models after decoding

5.2.4 MCEP models training and inference

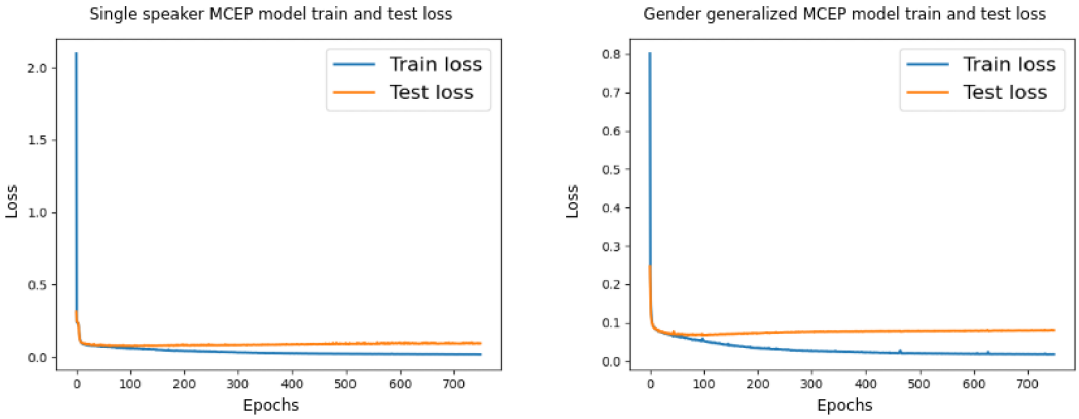

Figure 5.9: MCEP loss differences between single and multiple speaker models

The MCEP training has shown a steep decline in training loss especially on generalized speaker models, and after decoding and restoring the full dimensionality, the resulting predicted spectral envelope looked promising. However after evaluation phase, an increase of MCEPs was suggested, from 26 MCEPs to 50 MCEPs for both whispered and neutral speech, as the original method values seemed insufficient.

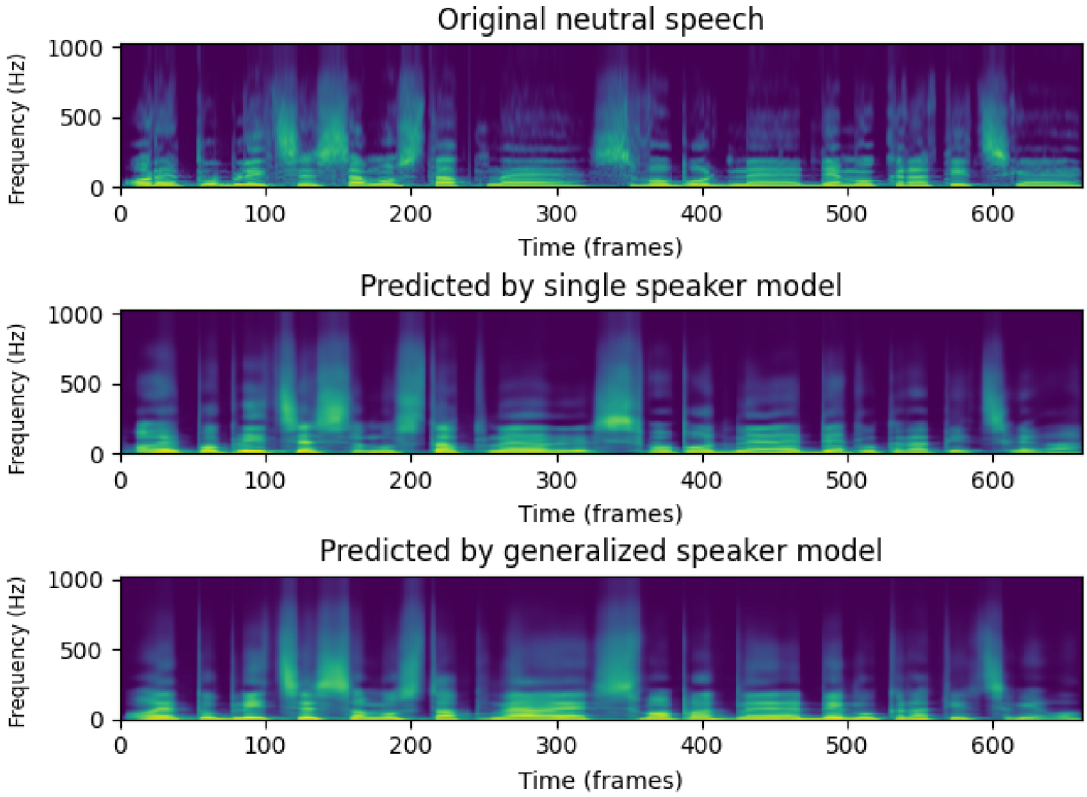

Figure 5.10: Inference comparison of MCEP models after decoding

5.3 Evaluation

Upon synthesizing converted waveforms from each speaker's test datasets using models trained on the corresponding speaker only, the BLSTMs have been able to replicate nearly all excitation and filtering trends of speaker's vocal tract. Surprisingly, using a gender generalized model on a dataset speaker worked comparably with the corresponding speaker single model. An important fact to note is that only a certain level of „naturalness" could be synthesized, since some of the data have been lost due to spectral envelope and aperiodicity coding.

However, using gender generalized models on out-of-dataset (or input) whispered utterances have shown limited results. One of the factors being the usage of varying input devices (microphones) working at different volume levels. Resulting synthesized audio was almost incomprehensible, so the MCEP number increase has been implemented, and has yielded better results in the terms of intelligibility, however, the outcome has still been far from desired „tailor-made" voice conversion.

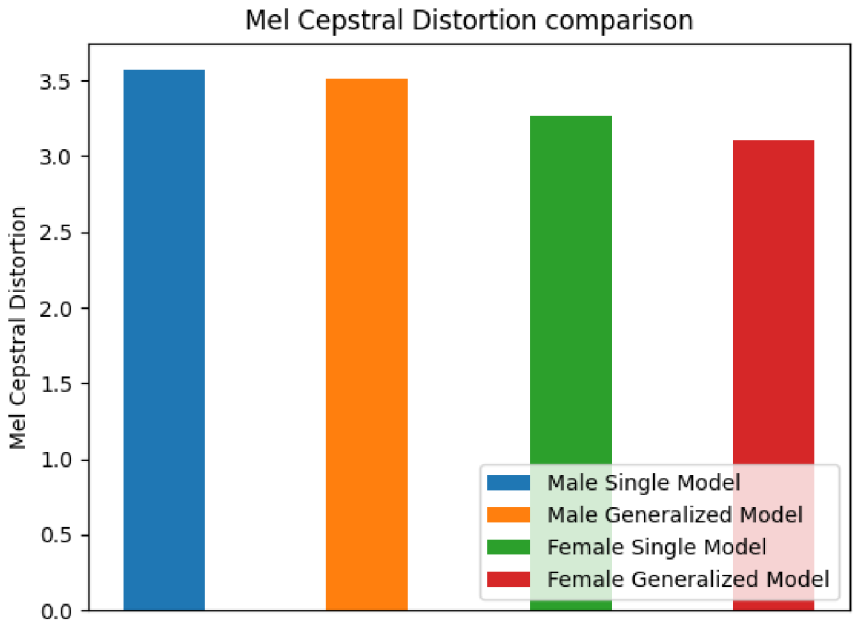

Figure 5.11: Mel-Cepstral Distortion differences between single-speaker and gender-generalized models

As can be seen in the figure 5.11, the male generalized model yields (surprisingly) slightly better results for in-dataset speaker, than „his own" trained model. Similar trend can be seen in the female models, however, the distortion is lower in general for both female models. That can be explained by the fact, that all 3 speakers had similarly high-pitched voices, whereas the male speakers voices differed marginally. Generally, lower MCD results mean better conversion. The MCD was calculated on the test sequences of the first speaker for each gender (meaning 10 utterances).

Chapter 6

Mobile application

The Android operating system has been selected as a host for the target application deployment. The development has been done in Java programming language, hosted on *Android Studio* [1]. The `min-sdk = 16` API requirement has been set, meaning any Android device hosted on older versions of the SDK platform could not run the application. However, the SDK guarantees support for 99,8% of the existing devices. The *MVVM (Model-View-ViewModel)* architecture has been used, separating the user interface from the application logic. Firstly, the user interface was implemented. The next step was implementing the inner application logic – the navigation between the UI fragments, storage and media access permissions and, finally, the recording method. At that point, the audio file (WAV format and 44100 Hz sampling frequency) was stored in the phone. At that moment, the conversion needed to be done, using the prepared python script and torch models. There, two courses of actions were possible. The first one was to deploy and execute the python code directly inside the Android OS. Such method has been enabled by several python-to-Java convertors and wrappers. One of those was *ChaquoPy* [2], which hosted a python interpreter inside the Java code and allowed importing dependencies and executing python code. However, the framework did not support all of the dependencies needed (namely PyWorld Wrapper), so the other course of action, a REST API, was selected.

Upon user request, the recorded whispered has been sent to a REST API hosted on *Heroku* [8] via a HTTP POST request. The implementation of the HTTP communication has been done using the *OkHttp3* [3] module. The hosting server has been implemented using *Flask* [4] and handles both POST and GET request methods. When the POST request with the file reaches the server, firstly a codec conversion is done in order to allow the script to load the file using SoundFile. The conversion is necessary, since Android uses the MPEG-4 container for audio media. That means that the *libsndfile* [4] based libraries cannot read the RIFF header of the file, therefore the conversion fails. To solve this issue, *ffmpeg* [5] multimedia framerowk is installed for codec support. Then, input file is converted to standard WAV format using *PyDub AudioSegment* [6].

[1] https://developer.android.com/studio
[2] https://chaquo.com/chaquopy/
[3] https://square.github.io/okhttp/
[4] http://www.mega-nerd.com/libsndfile/
[5] https://www.ffmpeg.org/
[6] https://github.com/jiaaro/pydub

Then, the whispered to neutral speech conversion is done on the server side using the same inference method used at evaluating the results of NN training i.e. WORLD whispered speech parameterization, calculation of deltas, DTW and feeding the data to the trained models. Obtained predicted neutral features are passed into WORLD for resynthesis and the resulting sound file is sent back to the user for playback.

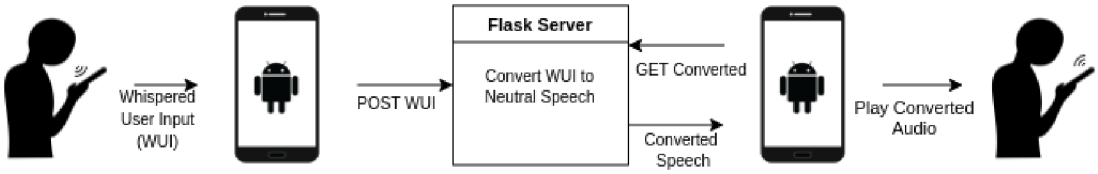

Figure 6.1: Android application outline

It is important to note, that the REST API is running on a free hosted environment, therefore has limited cababilities in the terms of computation power. That being said, when the conversion is taking longer than 30 seconds, the process is killed. This issue could be solved by upgrading server hosting. Therefore, it is advised that the input whispered utterance lengths should not exceed 5 seconds.

Chapter 7

Conclusion

The aspects of whispered speech analysis and voiced speech synthesis have been studied in this thesis. It follows up on the research article published by G. Nisha Meenakshi from Indian Institute of Science, Bangalore [7].

The emphasis is on on-demand near real-time conversion of whispered speech. A sizeable dataset has been acquired and further processed. BLSTM models have been trained, validated and tested on that dataset, for each individual speaker and for the gender generalized purposes. The synthesized output speech from inferenced models had been tested subjectively and the following results have been yielded: the converted whispered speech from the dataset speaker using single speaker model is clear, intelligible, and, in the boundaries of the vocoder (and feature coding), sounds naturally. The converted speech from dataset speaker using gender generalized model has yielded similar results, however, the synthesized pitch seemes to be „oscillating“ among the dataset speakers pitches. Lastly, converted whispered speech from out-of-dataset speaker using gender generalized model has returned limited results in the terms of intelligibility and those results have been highly dependant on the recording. The results are considerably influenced by the background noise, the microphone quality and the microphone placement. Some converted utterances can be understood, some are completely unintelligible. This issue could be fixed either by expanding the dataset and training in different acoustic conditions, or by data augmentation – augmenting the existing dataset – e.g. generating noise, adding reverb, etc.

The android application has been developed to employ trained BLSTM models for on-demand conversion. The conversion times are considerably dependant on hardware of the device that executes the conversion. When the conversion has been done on a local server, with high computational capabilities, the conversion took slightly more time than the utterance length. Which is, by definition, near real-time. When the conversion is done on the remote server with limited processing capabilities, the resulting conversion times may exceed utterance lengths multiple times. Conversion taking longer than 30 seconds is automatically suspended by server.

7.1 Future Works

The real time on-demand voice conversion is a trend with a rising popularity amongst users and researchers alike. Limited results of this thesis could be expanded by upsizing the dataset, using different parameterization and NN training techniques. One of such techniques could be an algorithm for defining the speaker's pitch, which could prove helpful

in removing the „oscillating“ pitch effect in the converted speech. Another addition could be a post-NN spectral envelope refinement using post-net [1](mentioned in [16]).

[1] https://github.com/NVIDIA/tacotron2

Bibliography

[1] Bäckström, T. *Introduction to Speech Processing* [online]. Aalto University, april 2019 [cit. 2021-04-18]. Available at: https://wiki.aalto.fi/display/ITSP/Deltas+and+Delta-deltas.

[2] Chevalier, G. LARNN: Linear Attention Recurrent Neural Network. *CoRR*. 2018, abs/1808.05578. Available at: http://arxiv.org/abs/1808.05578.

[3] Deshmukh, O., Espy Wilson, C., Salomon, A. and Singh, J. Use of temporal information: detection of periodicity, aperiodicity, and pitch in speech. *IEEE Transactions on Speech and Audio Processing*. IEEE. 2005, vol. 13, no. 5, p. 776–786. ISSN 1063-6676.

[4] Grinberg, M. *Flask web development: developing web applications with python*. „ O'Reilly Media, Inc.“, 2018.

[5] Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P. et al. Array programming with NumPy. *Nature*. 2020, vol. 585, p. 357–362. DOI: 10.1038/s41586-020-2649-2.

[6] Kawahara, H., Masuda Katsuse, I. and de Cheveigné, A. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds1Speech files available. See http://www.elsevier.nl/locate/specom1. *Speech Communication*. 1999, vol. 27, no. 3, p. 187–207. DOI: https://doi.org/10.1016/S0167-6393(98)00085-5. ISSN 0167-6393. Available at: https://www.sciencedirect.com/science/article/pii/S0167639398000855.

[7] Meenakshi, G. N. and Ghosh, P. K. Whispered Speech to Neutral Speech Conversion Using Bidirectional LSTMs. In: *Proc. Interspeech 2018*. 2018, p. 491–495. DOI: 10.21437/Interspeech.2018-1487. Available at: http://dx.doi.org/10.21437/Interspeech.2018-1487.

[8] Middleton, N. and Schneeman, R. *Heroku: Up and Running*. 1stth ed. O'Reilly Media, Inc., 2013. ISBN 144934139X.

[9] Morise, M. CheapTrick, a spectral envelope estimator for high-quality speech synthesis. *Speech communication*. Elsevier B.V. 2015, vol. 67, p. 1–7. ISSN 0167-6393.

[10] Morise, M. D4C, a band-aperiodicity estimator for high-quality speech synthesis. *Speech Communication*. 2016, vol. 84, p. 57 – 65. DOI:

https://doi.org/10.1016/j.specom.2016.09.001. ISSN 0167-6393. Available at: http://www.sciencedirect.com/science/article/pii/S0167639316300413.

[11] Morise, M. Harvest: A High-Performance Fundamental Frequency Estimator from Speech Signals. In: *Proc. Interspeech 2017*. 2017, p. 2321–2325. DOI: 10.21437/Interspeech.2017-68. Available at: http://dx.doi.org/10.21437/Interspeech.2017-68.

[12] Morise, M., Miyashita, G. and Ozawa, K. Low-Dimensional Representation of Spectral Envelope Without Deterioration for Full-Band Speech Analysis/Synthesis System. In: *Proc. Interspeech 2017*. 2017, p. 409–413. DOI: 10.21437/Interspeech.2017-67. Available at: http://dx.doi.org/10.21437/Interspeech.2017-67.

[13] Morise, M., Yokomori, F. and Ozawa, K. WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications. *IEICE Transactions on Information and Systems*. 2016, E99.D, no. 7, p. 1877–1884. DOI: 10.1587/transinf.2015EDP7457.

[14] Müller, M. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. 2015. ISBN 9783319219455.

[15] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché Buc, F. d', Fox, E. et al., ed. *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, p. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[16] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N. et al. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. *CoRR*. 2017, abs/1712.05884. Available at: http://arxiv.org/abs/1712.05884.

[17] Xu, Y., Li, Y. and Li, Z. Some Results on the Hadamard Product of Tensors. *Bulletin of the Iranian Mathematical Society*. Singapore: Springer Singapore. 2019081, vol. 45, no. 4, p. 1193–1219. ISSN 1017-060X.

[18] Zhang, S., Zheng, D., Hu, X. and Yang, M. Bidirectional long short-term memory networks for relation classification. In: *Proceedings of the 29th Pacific Asia conference on language, information and computation*. 2015, p. 73–78.

[19] Černocký, J. *Zpracování řečových signálů — studijní opora* [online]. FIT VUT v Brně, december 2006 [cit. 2021-04-18]. Available at: https://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf.

Appendices

Appendix A

Cookbook

A.1 Source Code and Libraries

Following list contains libraries and frameworks used for speech parameterization and neural network training. Newer versions might also work.

- Python => 3.9
- CUDA 10.2
- PyTorch 1.8.1 for neural network datasets, model training and inference
- NumPy 1.20.2
- SoundFile 0.10.3 for waveform read/write operations
- PyWorld 0.2.12 as a python wrapper for the WORLD vocoder
- mcd 0.4 for DTW
- Flask 1.1.2 for hosting cloud server on which the conversion is done
- PyDub 0.25.1 for server-side codec conversion
- Werkzeug 1.0.1 for flask server utilities.
- Java SE 11 for Android application development.
- OkHttp3 as an efficient HTTP client for android-server communication.

As per source code, the solution composes of consequent execution of python scripts, which are all dependent on the previous ones.

First, the silence removal script `sil_rm.py` needs to be used. After that, the feature extraction is done with `floader.py` script. Extracted features need to be padded using `sep_ftr.py` script and separated into three datasets. That is done using `train_test_sep.py`. The `lstm.py` and `dataTest.py` contain the definition of the NN datasets and models. Neural network training loop from `main.py` follows. The trained models are then saved, and can be inferenced with the last script – `test_models.py`.

The inference can be also done from already trained models using the `final_convertor` scripts. One uses the single speaker trained models, the other works with the generalized models.

A.2 Media Content

The attached media card consists of two directories: `app` and `conversion`. The application directory contains the `source` directory with all the source files for both the application and the server. The directory also contains an `app.apk` file, which is a packaged application file, which can be run on an Android device. The `conversion` folder consists of:

- models: trained neural networks for each speaker type.
- setup_new: subfolders with source files and speaker utterances for training the neural networks from scratch.
- source codes for conversion from already trained models.