Tampere University

Atte Virtanen

ROBUST F0 ESTIMATION OF TELEPHONY SPEECH USING ARTIFICIAL NEURAL NETWORKS

Faculty of Information Technology and Communication Sciences
Master of Science Thesis
February 2020

ABSTRACT

Atte Virtanen: Robust F0 estimation of telephony speech using artificial neural networks
Master of Science Thesis
Tampere University
Master's Degree Programme in Electrical Engineering
February 2020

The goal of this study was to develop a proper method for reliable estimation of fundamental frequency (F0) of telephony speech. This process of F0 estimation is often called "pitch tracking", and a large amount of research has been carried out on the topic. However, most of the studies have been performed on clean speech and robustness against adverse signal conditions is still an issue.

Today majority of recorded speech is from mobile phone calls. This poses two-folded requirement for robustness: First, speech is coded with a lossy speech codec modifying the audio wave form. Second, as a result of mobility, the human speaker is often located in a noisy environment, resulting in additive noise in the speech signal. Therefore robustness was the main topic of the study.

Traditional F0 estimation methods have been digital signal processing (DSP) -based, i.e., containing carefully defined mathematical operations and domain knowledge used only in algorithm design. However, late advancements in the field of Deep Learning makes it possible to let a computer learn the relevant representation of the speech signal in order to perform the F0 estimation task. This approach was taken in the present study.

A starting point for the study was an F0 estimation algorithm developed in Aalto University [10]. The present thesis builds on top of that work by incorporating speech codecs in model training, and evaluating the resulting system with clean and coded test data. The use of speech coded signals in training phase was motivated by the fact that a learning method should learn from data having similar properties as the evaluated data. On the other hand, speech coding can also be seen as a data augmentation method to increase model robustness even with non-coded speech signals.

The proposed method with different training strategies was compared to a number of established DSP -based methods. Comparison was done with all combinations of speech coding and noise scenarios. The results show that the proposed method clearly outperforms traditional methods when background noise is present in coded speech. A slight performance improvement was also achieved with clean speech signals, indicating that the proposed method should always be used instead of DSP -based method unless computational resources are of scarce.

Keywords: F0, AMR, deep learning, pitch tracking, speech analysis

TIIVISTELMÄ

Atte Virtanen: Puhelinpuheen luotettava perustaajuuden estimointi neuroverkkojen avulla
Diplomityö
Tampereen yliopisto
Sähkötekniikka, DI
Helmikuu 2020

Tämän tutkimuksen tavoite oli kehittää menetelmä puheen perustaajuuden (F0) luotettavaan estimointiin puhelinpuheesta. Kirjallisuudessa tästä puheen perustaajuuden estimointiongelmasta käytetään usein termiä englanninkielistä termiä "pitch tracking"ja aiheesta löytyykin paljon tutkimustietoa. Suurin osa tutkimuksesta on kuitenkin suoritettu puhdasta puhetta käyttäen ja tulosten yleistyminen erilaisissa olosuhteissa tallennettuun puheeseen kaipaa lisätutkimusta.

Nykyisin suurin osa tallennetusta puheesta on kulkenut matkapuhelinverkossa eli ainakin toinen päätelaite on ollut matkapuhelin. Tästä on seurauksena kaksi haastetta puheen analyysissä. Ensinnäkin matkapuhelinverkossa siirtoa varten puhe koodataan häviöllisesti eli tietoa kadottaen. Toiseksi puhuja voi olla meluisassa ympäristössä ja siten puheen seassa saattaa olla additiivista kohinaa. Tämän vuoksi tutkimuksessa keskityttiin erityisesti käytettävän menetelmän robustisuuteen eli häiriösietoisuuteen.

Perinteiset menetelmät perustaajuuden estimointiin ovat perustuneet digitaalisen signaalinkäsittelyyn (DSP, Digital Signal Processing) eli matemaattisesti määriteltyihin operaatioihin, joiden avulla ja sovellusalue tuntemalla on mahdollista tutkia signaalin sisältämää informaatiota. Viimeisten vuosien aikana tapahtunut edistys syväoppimisen tutkimuksessa on mahdollistanut järjestelmät, joissa tietokoneohjelma itse oppii tunnistamaan signaalista piirteitä ja niiden perusteella päättelemään asioita, esimerkiksi tässä tapauksessa signaalin sisältämän puheen perustaajuutta. Syväoppimiseen perustuva puheen perustaajuuden estimointi valittiin työn lähtökohdaksi ja perinteiset menetelmät toimivat verrokkeina.

Tutkimuksen lähtökohtana toimi Aalto-yliopiston tutkimusryhmässä kehitetty keinotekoisiin neuroverkkoihin perustuva algoritmi [10]. Tämä työ yleistää algoritmin käyttöä puhekoodekin kautta kulkeneelle puheelle lisäämällä algoritmin opetusmateriaaliin puhekoodattua puhetta ja vertaamalla suorituskykyä muihin menetelmiin niin koodatulla kuin koodaamattomalla puheella. Koodatun puheen lisääminen neuroverkon opetukseen on perusteltua, sillä neuroverkko voi toimia varmasti oikein vain sellaisella datalla, jonka ominaisuudet se tuntee. Tämän lisäksi oletuksena on, että koodattu puhe toimii myös datan lisääjänä ja siten parantaa myös neuroverkon yleistämiskykyä.

Ehdotettua neuroverkkomenetelmää erilaisilla koulutusvaihtoehdoilla verrataan useisiin tunnettuihin DSP -pohjaisiin menetelmiin. Vertailu tehdään kaikilla signaalityyppien yhdistelmillä eli eri koodekit ja kohinatapaukset käsitellään erikseen. Tulokset osoittavat, että ehdotettu menetelmä suoriutuu merkittävästi perinteisiä menetelmiä paremmin, mikäli puhesignaali sisältää paljon taustakohinaa tai on puhekoodattu.

Myös puhtaan puheen tapauksessa ehdotettu menetelmä suoriutui hieman verrokkeja paremmin, ja on siten tarkkuuden puolesta soveltuva kaikkiin puheen perustajuuden estimointitehtäviin. Mikäli laskentaresurssit ovat rajatut tai aikaa algoritmin kouluttamiseen ei ole, verrokkina olleet perinteiset menetelmät ovat vielä parempi vaihtoehto.

Avainsanat: F0, AMR, syväoppiminen, puheen perustaajuus, puheanalyysi

PREFACE

This thesis is part of Finnish Academy funded project "Rhythms in Infant Brain: Wearables for Computational Diagnostics and Mobile Monitoring of Treatment (RIB)" and prepared in co-operations between Aalto University Department of Signal Processing and Acoustics and Tampere University Signal Processing Research Center.

My deepest gratitude goes to my supervisor Prof. Okko Räsänen who not only gave me the opportunity to finish my prolonged studies and also made it happen with invaluable advice, push and support. DSc. Manu Airaksinen provided the core of the thesis with the method developed by him and guidance how to extend its application and evaluation to another domain.

The direct technical guidance provides a solid base to the research and thesis work but, of course, other kind of support cannot be overestimated. I want to thank my room mates Anant and Khazar for being there when I was struggling with my work. Thanks to Prof. Tuomas Virtanen for creating an inspiring atmosphere around audio signal processing at Tampere University.

There are so many colleagues both from the current thesis period and previous professional and studying life worth for various reasons worth naming here that I can only say Thank You to all of you, you know who you are and I wish we still meet again.

Above all, I need to thank the most important people in my life, my family. My beloved wife Jenni has all these 15 years supported and pushed me with the graduation project. I was not an easy ride but will end soon. Thank you! Our kids Kerttu and Lauri are the foundation of our life and gave the reason to go through this project. Obviously my parents are also to thank, for everything!

Tampere, 3rd February 2020

Atte Virtanen

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PROGRAMS AND ALGORITHMS

LIST OF SYMBOLS AND ABBREVIATIONS

AMR-WB	Adaptive Multi-Rate Wideband, a wideband speech codec used in mobile networks
AMR	Adaptive Multi-Rate, a speech codec used in mobile networks
ANN	Artificial Neural Network
API	Application Programming Interface
CELP	Code-Excited Linear Prediction, method for speech coding
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DSP	Digital Signal Processing, here: traditional methods composing of basic mathematical operations not including machine learning
EGG	Electroglottogram, a non-intrusive method for vocal fold position tracking
F0	Fundamental frequency
FFE	F0 Frame Error
FIR	Finite Impulse Response, a digital filter type
FPE	Fine Pitch Error
GCI	Glottal Closure Instant, the moment of closed glottis during voiced speech
GPE	Gross Pitch Error
GPU	Graphics Processing Unit, microprocessor type originally developed for graphics calculations but also used with ANN calculations
GRU	Gated Recurrent Unit, Recurrect Neural Network subtype
LP	Linear Prediction, operation used in many speech processing applications
LSTM	Long Short-Term Memory, Recurrent Neural Network subtype
LTE	Long Term Evolution, Mobile network standard
ML	Machine Learning
NB	Narrowband, speech sampled at 8000 Hz
NCCF	Normalized Cross-Correlation Function
PCM	Pulse Code Modulation

RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
STFT	Short-Time Fourier Transform, a method for transforming a part of time domain signal to frequency domain
VDE	Voicing Decision Error
WB	Wideband, speech sampled at 16000 Hz

1 INTRODUCTION

Speech is perhaps the most distinctive feature of humans among other animals. Communicating via sound waves is a common feature in nature, but simple sounds alone do not carry enough information to express complex entities. By development of language and versatile speech production system, humans achieved such a level in communication that it was possible to express arbitrary things by voice.

The ability to speak made it possible accumulate information by spreading it mouth to mouth. Before invention of writing this was the only true way to develop civilization, so that not every generation had to start from scratch and personally gain the experience required for survival. Now parents could teach their children how to act in new situations and the wisdom of elder people increased their value even if they physically were already a burden to community.

Language is not the only dimension of speech. In addition to words, speech carries a vast amount of information regarding the speaker itself and the intention of spoken words. Human voices are quite distinguishable, and people can recognize a familiar speaker without seeing or hearing the direction of incoming speech. This is a very important aspect of communication in communities.

Speech can be varied to transmit feelings or urgency. Depending on intonation, stressing and other properties, semantically identical utterance can have different meaning, for example place an imperative or question form on top of declarative meaning. Therefore machine listening at broad is a much more complex issue than just traditional automatic speech recognition transcribing audio waveform into text.

Fundamental frequency, or F0, is one of the basic properties of speech. A large part of speech is voiced, meaning that it has a quasi-periodic structure. This periodicity comes from vibrating vocal folds which modulate the airflow coming from lungs. Frequency of the vocal fold vibration is affected by many linguistic and neurophysiological factors, hence carrying a lot of information regarding the speaking style or, e.g., health or mood of the speaker. Therefore robust estimation of F0 is also required in several uses cases ranging from basic speech research (e.g. [33]) to applications such as speech synthesis or speech-based diagnostics of speaker health [16] or occupational voice (e.g. [36]). F0 is also one of the standard features included in the speech analysis toolkits OpenSMILE [17] and Praat [12] which are widely used in speech processing tasks.

In this thesis we provide an overview of current state of F0 estimation methods and pro-

vide a new method to further improve the robustness of F0 estimation. Usually F0 estimation methods have been developed with clean speech signals and possibly somehow verified with noisy or otherwise imperfect signals, or, alternatively, designed specifically for noisy conditions using dedicated signal processing mechanisms. Our approach is to use machine learning to create an F0 estimation model that learns to cope with disturbances already in the algorithm training phase. The current work is based on the method initially reported in [10]. We improve the robustness of the original algorithm in telephony applications by applying speech coding to the training data in our training procedure.

This thesis is organized as follows. In chapter 2 we introduce basics of human speech and also the properties related to speech coding. In chapter 3 estimation of the F0 is discussed. We introduce the main principles used in F0 estimation and some of the present algorithms. In chapter 4 we take a look at artificial neural network (ANN) -based estimation along with introduction to ANNs. We also present our convolutional neural network -based F0 estimator that is inspired by the WaveNet architecture originally developed for speech synthesis [40]. In chapter 5, our test and evaluation setups are described in detail. Chapter 6 presents the results obtained. An overall wrap-up, significance of the work done, and possible future improvements are discussed in chapter 7.

2 HUMAN SPEECH PRODUCTION

Speech production is very complex entity starting from abstract thought and ending to acoustic pressure waves radiating from mouth and nose. Between these, neural and muscular operations take place, converting the communication intention to movements of the parts of the speech production system, also referred to as articulators. In this chapter, we consider only the last part, the mechanical production of speech.

2.1 Speech production system

As speech is a form of acoustical energy which in turn is (air) pressure variation in time, the first thing in speech production is to generate air pressure. This takes place in the lungs with the aid of diaphragm and abdominal muscles, pressing air through upper parts of the respiratory (and vocal) tract. Without any interaction of speech producing organs, the resulting action would be breathing and the sound would be just noise of flowing air. Actual speech is formed in phonation and articulation processes taking place in larynx and vocal tract.

2.1.1 Larynx and phonation

Actual generation of speech starts to take place in larynx area. Figure 2.1 shows the cross-sectional view of human speech production organs. Air always flows through larynx and vocal folds, but by controlling the position and tension of the vocal folds, the speaker can create voiced sounds.

A voiced sound is periodic and it is generated when vocal cords vibrate, corresponding to periodic opening and closing of the glottis, which is the space between the vocal folds. As a result, the constant airflow from the lungs changes into a process in which accumulating subglottal pressure first opens the glottis with increasing amounts of airflow through it, followed by a sudden closure of the glottis due to the suction caused by the rapid airflow through the gap (also known as the Bernoulli effect). This is again followed by pressure accumulation and opening of the glottis. The rate of this cycle is dependent both of the air flow from lungs and the tension on vocal folds. Result is an air flow with pulses. This periodic air flow change is called phonation. The inverse of the period of pulses is the fundamental frequency, or F0, of speech.

The frequency of the pulses is not the only characteristic of the phonation. It can be further categorized to three types: breathy, modal or pressed phonation based on the shape of the pulse. This can be actively varied by the speaker through muscular control of the vocal fold tension, or can be caused by different non-intended mechanisms such as pathology to the vocal folds. One special case is whispering, where vocal folds are not vibrating at all, and where also typically voiced speech is presented as unvoiced.

These aforementioned phonation types are presented in detail in figure 2.2. First is presented the cross-section of larynx to demonstrate the glottis opening. Second figure in each case presents glottal excitation obtained from speech signal by inverse glottal filtering. The third figure presents the spectrum of the glottal excitation. In case of breathy phonation (figure 2.2a) vocal fold tension is light and less pressure is needed for opening glottis. This results in wide pulse in time and less high frequency components in spectrum. Pressed phonation (figure 2.2c) on the other hand is result of tense vocal folds and resulting excitation is shorter and sharper pulses in time having wider range of frequency content. Typical modal phonation (figure 2.2b) is between aforementioned extreme cases.

In general, phonation takes place when vocal folds are tight and glottis is small enough to allow pressure accumulation. For unvoiced speech, glottis is wide open in a similar way as when breathing and air flows freely through the glottis.

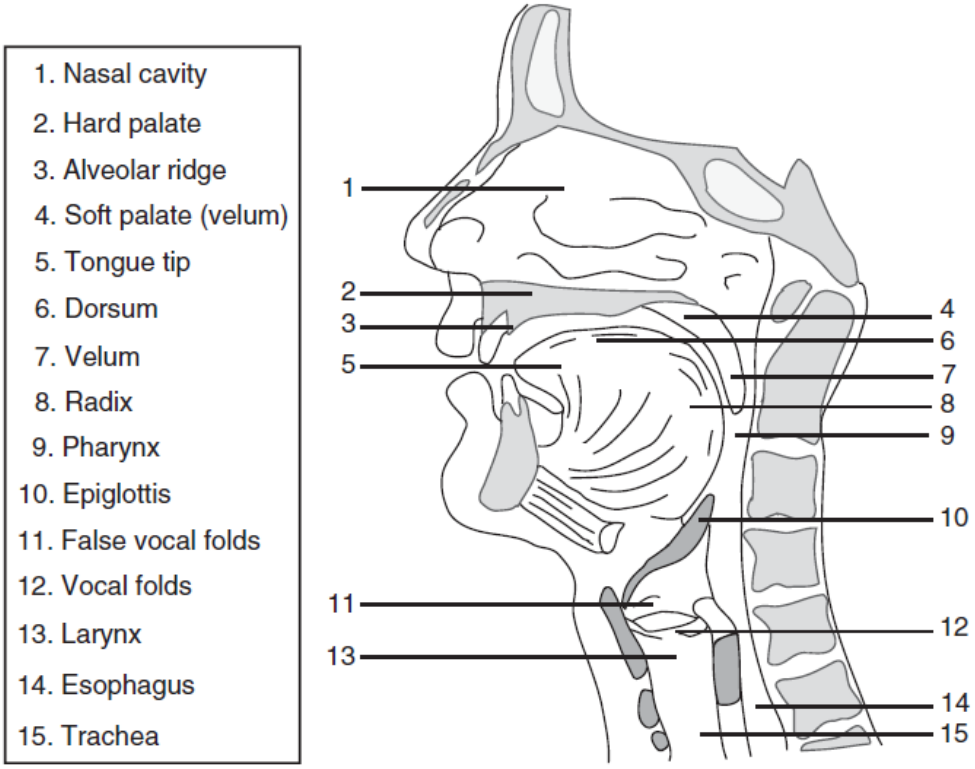


Figure 2.1. *Human speech production mechanism. Reproduced from [44]*

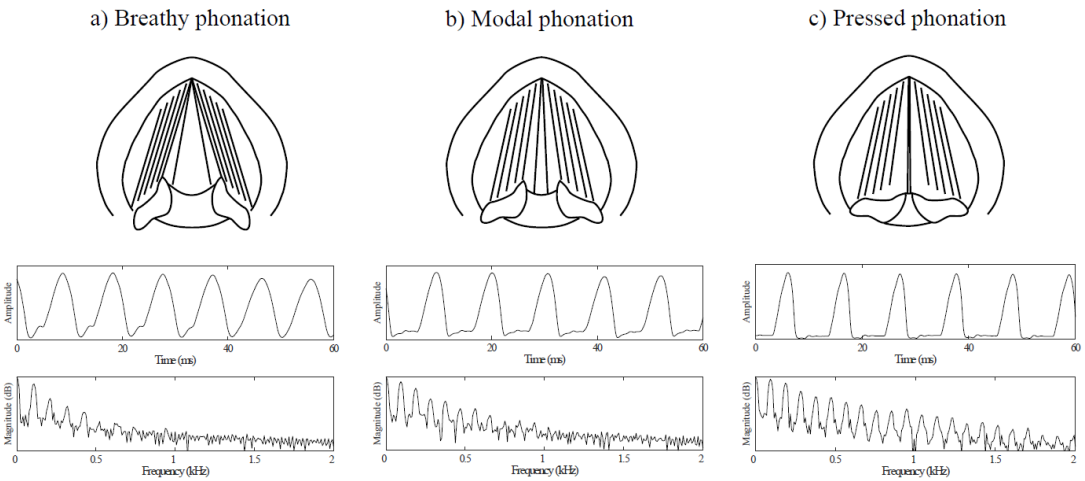


Figure 2.2. *Different phonation types. Topmost figure depicts glottal vocal fold configuration, middle figure resulting glottal excitation and bottom figure excitation spectrum. Printed with permission from [8]*

2.1.2 Vocal tract and nasal tract

The airflow from the larynx is often referred to as the excitation signal, which is either periodic for voiced or noise-like for unvoiced speech. This excitation signal is not very expressive yet, and it needs to be shaped to produce more diverse sounds corresponding to different speech sounds of the given language. This shaping process is called articulation, and it takes place in the vocal and nasal tracts.

At pharynx there is a flap called velum which controls the airflow division between oral cavity and nasal cavity. Nasal tract has a fixed form, thus letting the air flow freely out from nose. This means that alone it cannot produce different sounds, but it is used in addition to oral path to create more diverse set of speech sounds.

Most of the articulation process takes place in the mouth. Oral cavity acts as a resonator and articulators shape the vocal tract. Tongue, jaw, and lips are the most influential organs is articulation. They shape the vocal tract, thereby effectively adjusting the frequency response of the channel. This introduces *formants*, which are the resonance frequencies of the vocal tract. The locations of the formants in the frequency space mostly characterize voiced speech. Whispering is a special case where all the normally voiced phones are also produced as unvoiced, thus vocal tract alone determines the identity of the phones.

Some unvoiced and mixed sounds are also characterized by formants. Most extreme vocal tract cases contain either very narrow gap or even full closure of vocal tract. Narrow gap, or constriction, causes high pressure and thus high velocity air flow, producing turbulent flow and noise-like sound. This is called frication. Some phones require a full closure of the tract to create a transient sound. These sounds, called plosives, are generated when the vocal tract if fully closed for a short period of time to build up pressure,

followed by sudden release of it.

2.1.3 Lip radiation

The final stage of speech production is related to the release of the produced air flow variations from the oral and nasal cavities to the external environment as acoustic air pressure variations. The air flows between lips, which not only act as a part of vocal tract, but as a radiator with directional properties. Low frequencies with wavelength exceeding head dimensions are radiated relatively omnidirectionally. At higher frequencies, clear directivity is observed [44]. In addition, the external environment acts as an acoustic impedance mismatch with the vocal tract described in detail in e.g. [34]. More broad overview can be found in e.g. [9] and the conclusion is that lip radiation effect for lower frequencies can be considered as a first order differentiator with a 6dB/octave emphasis on higher frequencies. As a result of these factors, high frequencies are amplified compared to lower frequencies, compensating for part of the low-pass characteristics of the glottal excitation.

2.2 Linguistic properties of speech

Speech and language can be seen as having a hierarchical structure. The smallest unit in speech is a *phone*, which is the smallest speech sound that can be separated from speech. Close match to the phone on a theoretical level is a *phoneme*, which is more language specific, and where phonemic contrast is defined as the minimal change in the speech sound that can lead to a change in meaning in the given language. Phones are true speech parts and phonemes more theoretic entities but sometimes they are used interchangeably in the literature.

Phones (or phonemes) form rhythmic units called *syllables* which are still mostly meaningless, except for monosyllabic words like "dog". Syllables are building blocks of *words*, which are the smallest units with an individual meaning. An *utterance* is a group of words in speech delimited by breath or other pauses. They may or may not convey a complete meaning to be commmunicated, which is the property of a *sentence*.

An example of the acoustical waveform of an utterance is presented in figure 2.3. It is part of the PTDB-TUG database [42] and extracted from file mic_F01_sa1.wav.

A more detailed view of a single word is presented in figure 2.4. It is an excerpt from the utterance in figure 2.3 and presents the word "suit". This word clearly has three parts: [s] (roughly 1.25-1.4 s), [u] (1.4-1.55 s), [t] (1.6-1.7 s). [s] is a typical fricative consonant with a noise-like signal structure. [u] is voiced vowel with very clear periodicity. [t] is a plosive first having a stop period followed by a noise-like burst.

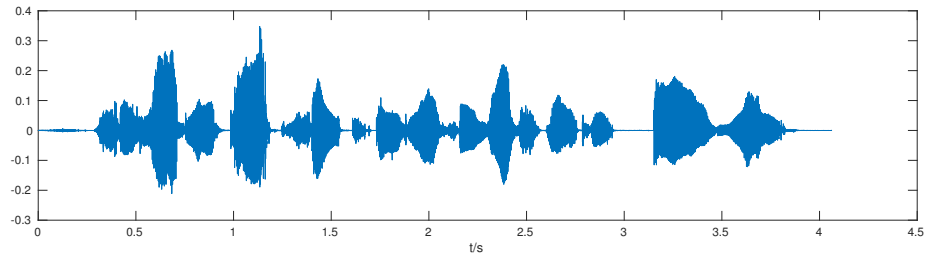


Figure 2.3. *Waveform of female spoken utterance "She had your dark suit in greasy wash water all year."*

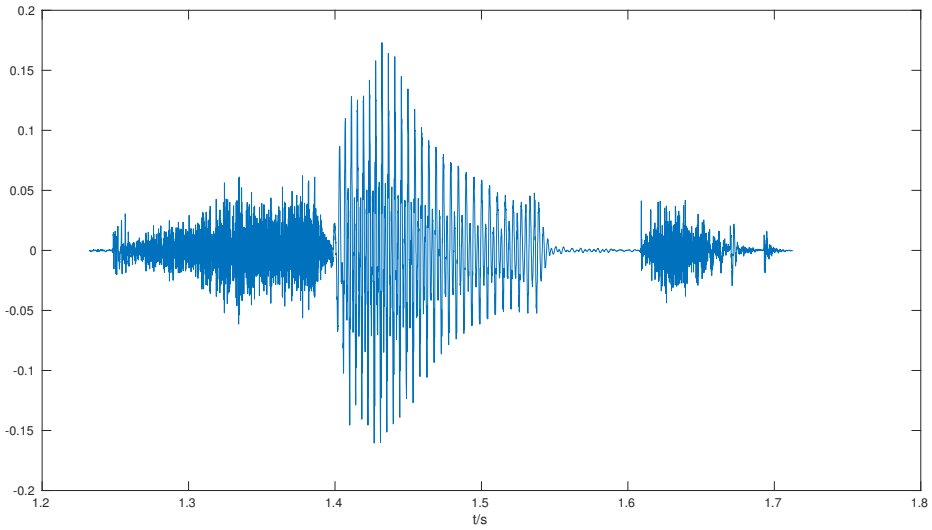


Figure 2.4. *Waveform of a female spoken word "suit"*

2.2.1 Modeling of Human speech production

One rather simple way to model the speech production system is to split it into two blocks, one for the excitation signal and another for vocal tract and lip radiation. This is called source-filter model. Excitation signal block generates the basic waveform, such as periodic signal for voiced speech or noise for mimicking turbulent airflow of fricatives. Vocal tract part is a linear filter, usually all-pole, which shapes the excitation's frequency response. The block diagram is presented in figure 2.5. Even though this model is very much simplified, it is broadly used in numerous applications, such as speech coding or speech feature extraction [45].

2.2.2 Speech coding

Most speech codecs apply the aforementioned source-filter model because of its mathematical simplicity. The whole system can be expressed as a single transfer function which in turn is very easy to implement with digital filters. An important note here is that efficient

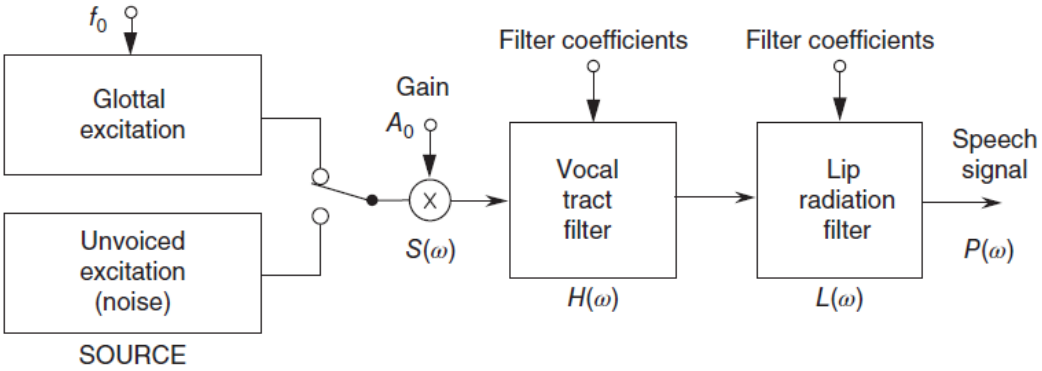


Figure 2.5. *Block diagram of source-filter model. Reproduced from [44]*

speech coding is lossy and perceptual, i.e., for human listener most of the information is preserved but the actual waveform can be seriously distorted from the original uncoded speech. This is illustrated in figure 2.6 where the same word is presented as an original waveform sampled at 16 kHz and as the corresponding typically used 12.65 kbps AMR-WB decoded waveform. The original signal is sampled at 16-bit accuracy, corresponding to an overall bitrate of 256 kbps, which is roughly 20 times the bitrate of the encoded signal. Perceptually the signals are very close to each other, only the beginning [s] starting at 1.25 s and ending at 1.40 s is muffled in AMR-WB coding but the voiced part is very close to the original. The highest frequencies (above 6400Hz) are not coded at the 12.65 kbps bit rate, which partially explains weak frication energy in [s].

Modern speech codecs such as AMR are using CELP-like analysis-by-synthesis structure originally presented in [39]. Excitation is modelled with a fixed codebook of signal vectors while vocal tract parameters are obtained by linear prediction (LP) analysis. Analysis-by-synthesis codecs create the encoded speech by synthesizing a number of signal candidates and choosing the best one out of those and passing the corresponding parameters to a decoder. LP coefficients are straightforward to calculate and the amount of resulting data is small. Most of the bits in the encoded speech are consumed by source signal, corresponding to the index of the vector quantized codebook. Finding the optimum vector is time consuming and a full exhaustive search is not feasible in real-time communication. Smart algorithms have been developed for the search, and "good enough" outcome can be achieved with acceptable computation times in practical hardware such as mobile phones.

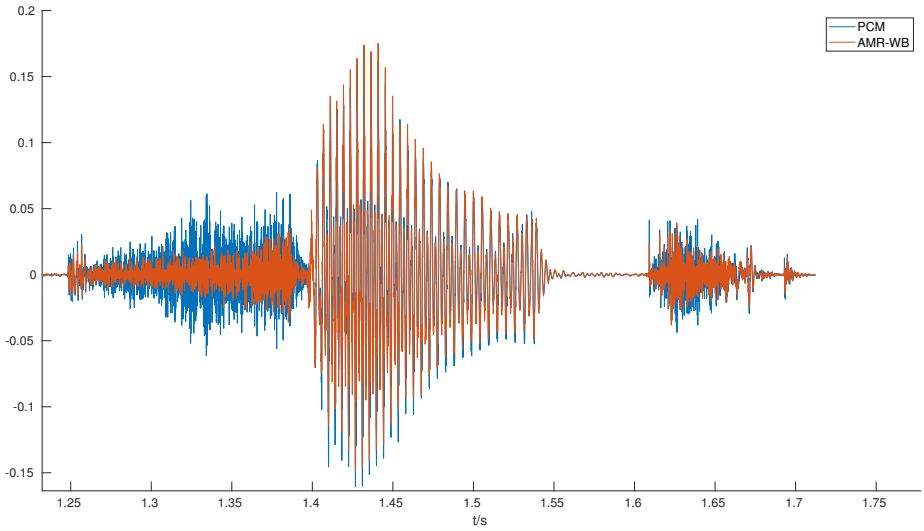


Figure 2.6. *Waveforms of female spoken word "suit" as raw PCM and AMR-WB coded*

3 F0 ESTIMATION

The fundamental frequency is defined to be the one with lowest period in periodic signal of interest, and in speech it corresponds to the vibration frequency of the vocal folds. In speech signals, fundamental frequency really is not the only periodic component. Due to the human speech generation mechanism, harmonics of the fundamental are also always present. That property poses a challenge for f0 estimation, since signal has several periods and fundamental might not be the strongest one.

The definition of perfect periodicity for a signal is

$$x(t + \alpha T) = x(t) \tag{3.1}$$

where x denotes signal of interest, t time, α is an arbitrary integer ≥ 1, and T period length. Here we want to estimate the value of T corresponding to fundamental frequency. It is to be noted that natural speech is not in reality perfectly periodic, but *quasi-periodic*. I.e., the consecutive fluctuations of the vocal folds have *jitter* in their period lengths. In addition, the period changes as a function of time due to intonation and disappears totally when voicing ends. Nevertheless, the aforementioned model of perfect periodicity is a commonly applied simplification in F0 estimation, and the assumption is only applied to short segments of signal at a time.

3.1 The problem of F0 estimation

The first problem in determining F0 of speech is to decide if there is speech at all and is it voiced or not. This voicing decision is crucial part of successful estimator, since F0 is defined only for voiced phones. Unvoiced speech has no real F0 and can be omitted. Usually, voicing decisions are based on the same algorithm-internal mechanics that are also used to determine the value of the F0. As an example, autocorrelation-based F0 estimation methods usually attempt to measure the height of the dominant peak in the signal autocorrelation curve as a proxy for the strength of voicing, while the temporal delay at which the peak occurs corresponds to the period of the F0.

Although F0 is a well-defined property, its estimation is not always straightforward. One reason to this is the fact that F0 is usually estimated from speech signal which is a product of a cascade of operations in human speech production system. Glottal pulses are

generated in an early phase of the pipeline, so final acoustic speech signal only weakly resembles the original glottis signal in terms of its periodical characteristics. Another big issue is that speech is analyzed in frames containing multiple F0 periods (or glottal pulses). It is assumed that the interval between consecutive glottal pulses is the same (i.e., F0 stays constant within the analysis window), but in reality it seldom is constant even for short periods. Therefore the F0 estimate can either reflect the average of several glottal pulses, or be biased towards strongest pulses in the analysis window.

Due to these challenges, straightforward DSP operations are not necessarily enough to provide reliable F0 estimates. One needs to pre-process the speech signal first to remove extra distractors such as the DC component. Even after the pre-processing stage, common periodicity estimators such as autocorrelation and cepstrum (introduced in 3.2) usually end up in several F0 candidates. Post-processing is then applied to select the best candidate as the F0 estimate. This stage often contains heuristics which are often driven by hand-designed rules on how to act in certain kind of situations, or using information from multiple neighboring frames to find a solution that fits to the current temporal context as F0 can only change from frame-to-frame with a finite speed. This post-processing area is covered in more detail in Section 3.2.5.

All of the above applies to clean recordings of speech. In practice, high quality recordings are usually only available in laboratory situations i.e. with low levels of reverberation and background noise, whereas most real life applications and naturalistic data collections will take place outside controlled environments. Usually any recorded speech signal contains some amount of additive noise either from the recording device (with possibly known properties) or environmental noise with random properties. Typically lower signal to noise ratio (SNR) leads to worse results in any signal analysis. In this case especially a competing speaker, transient noises, or any other noise with speech-like characteristics may severely affect the F0 estimation performance.

Another kind of source of distraction is channel variability, mathematically meaning convolution of the original signal with some kind of transmission channel that depends on the use context. The most typical case is room reflections and reverberation in general. Reflected speech reaches the microphone later than the direct speech and is probably also spectrally altered. Technically speaking this is about impulse response of the system being different from an ideal Dirac delta pulse. These kind of noise scenarios have been analyzed and tested with F0 estimators in, e.g., [21] and [30].

Additive noise and channel noise are results of linear phenomena. Many audio processing techniques are non-linear and usually non-linear problems are harder to solve than linear. A substantial proportion of speech data transmitted or stored for analysis is from telephone calls or video recordings.This also means that they are most likely processed by an audio codec in order to reduce the size of the data. Lossy codecs discarding perceptually non-relevant data may alter the speech signal nonlinearly and even change the frequency content. This is especially true for low bit-rate speech codecs which are audio codecs optimized for speech. They do very heavy processing to audio waveform, which

may be perceptually not so different from the original but have a drastic difference in the resulting signal waveform.

3.2 Basic methods for F0 estimation

Periodicity estimation can be done both in time and frequency domains. In principle, time domain methods utilize the original signal as a sequence of samples within an analysis window, whereas frequency domain methods are based on a spectrum usually created by short time Fourier transform, also applied to a windowed signal. The time domain approach typically includes finding the longest period using the (auto)correlation function. In the frequency domain, the spectrum shows periodical components as evenly spaced peaks, and F0 is estimated from those. Typically processing is done in overlapping windows (also called frames), which means that only a small part of the signal is analyzed at a time and successive frames partly contain the same data. In practice the reported F0 for a specific time instant is some sort of average inside the frame centered at this time instant.

Some methods may not directly seek the fundamental frequency inside frame and processing may not be based on frames at all. Example of this kind of processing is REAPER [50] algorithm which actually estimates GCI (glottal closure instant) track and calculated F0 estimates based on that.

Figure 3.1 shows an example of a frame containing voiced speech. The frame length is 512 samples and sampling frequency 16 kHz which corresponds to 32 ms. Periodicity is well visible and should be easily estimated. In Figure 3.2 frame signal is mostly non-voiced. The first 10ms look like periodic but the rest of the signal is noise-like. This frame would most probably be considered non-voiced.

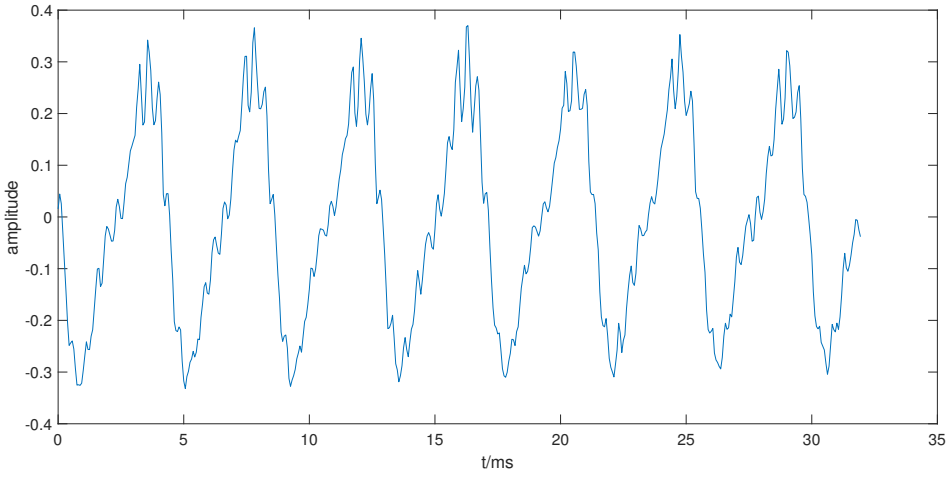


Figure 3.1. *Example frame of voiced speech*

Here we present a few key methodologies which are also sometimes used as basic blocks

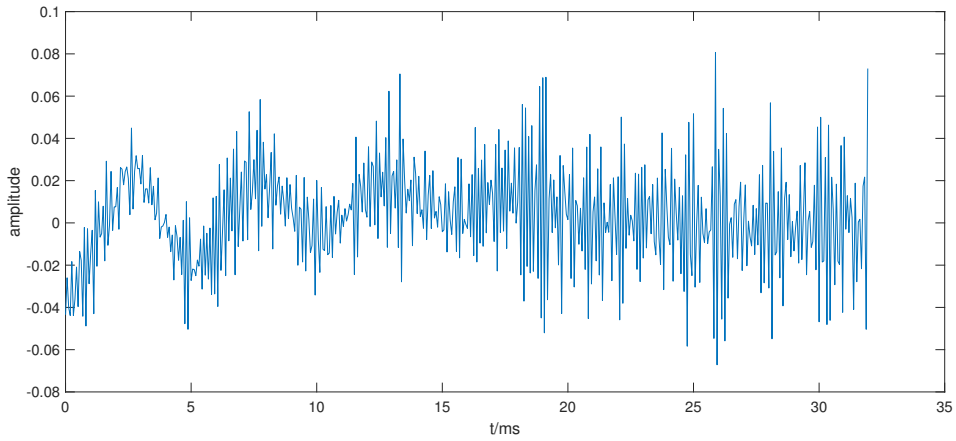


Figure 3.2. *Example frame of mostly non-voiced speech*

in several more advanced F0 estimators. These are generic operations suitable for many other estimation tasks. As such, none of these are sufficient beyond the most simple periodicity estimation problems, as they propose F0 candidates out of which the best one should be picked in a separate post-processing phase (Section 3.2.5).

3.2.1 Autocorrelation

As was stated in Section 3, a periodical signal repeats itself. Knowing this property, one of the first things to try is estimating this period directly from the time series signal. This is mathematically done by a correlation function which is maximized when correlated signals are maximally similar. When we are estimating the similarity of a single signal against itself at different relative delays, the operation to be used is referred to as *autocorrelation*. It is formalized as

$$R_{XX}(k) = \sum_{k=-K}^{K} x(n) * x(n-k) \tag{3.2}$$

where x is the signal, k is the lag and K maximum lag corresponding to the lowest potential F0 of interest. Sometimes autocorrelation coefficients are normalized between [-1, 1] with the maximum value of one at zero-delay, thereby corresponding to the Pearson correlation coefficient.

Maximum value of the autocorrelation function tells at which lag the signal repeats itself most accurately. Due to the problems mentioned in 3.1, this might not be the true F0 lag but either a multiple or fraction of it. Further actions need to be taken to select the most probable candidate as an estimate. Actual F0 frequency is then calculated from lag of k

samples as

$$F0_{est} = \frac{1}{k} * Fs \tag{3.3}$$

where Fs is the sampling frequency.

Autocorrelation has its maximum value at a lag of zero samples which also represents the signal energy as 3.2 reduces into sum of squared sample values. F0 candidates are the strongest lags within a non-zero lag search range that spans the minimum and maximum desired fundamental period lengths.

Autocorrelation is a simple and traditional method for obtaining F0 candidates. It is easy to implement and is fairly robust to noise if long analysis windows are used. Computational complexity is proportional to window length and all possible F0 candidates can be found with a single correlation operation.

The main shortcoming of the autocorrelation operation is the limitation of having to set the correlation window to be at least the expected maximum glottal period corresponding to the lowest expected F0. This means long correlations that result in large numbers of calculations and, more importantly, the resolution for estimating single glottal periods is lost since multiple high-pitched periods can exist within a single window. Also a rapid change in F0 smears the peak in the autocorrelation function. Finally, even if F0 was constant during the entire window, the confidence of the estimate depends on the value of F0, since more periods of high F0 fit into fixed-length windows [49].

3.2.2 Cross-correlation

In the autocorrelation function, exactly the same signals are correlated. Cross-correlation is a more general operation, where the signals under investigation are arbitrary. When estimating the F0, the cross-correlation is calculated between an analysis window of the signal of interest and a delayed part of it. Cross-correlation is separately calculated for each possible delay and window length corresponding to a specified F0 range. This approach fixes the two problems of the autocorrelation mentioned in 3.2.1: Single glottal periods can now be estimated with the adaptive-length windows, which solves the problem of variable amounts of periods on a fixed-length frame. Also, a similar resolution of periodicity confidence is achieved across the desired F0 search range. [49]

Cross-correlation is defined as:

$$R_{XY}(k) = \sum_{k=-K}^{K} x(n) * y(n-k) \tag{3.4}$$

where x is the actual signal, y is candidate period signal of x and K lag of the candidate period.

As the window length is varying with each F0 candidate, the maximum signal energy is too. Therefore normalization is essential for getting varying length correlations comparable. Here we introduce normalized cross-correlation function (NCCF) which normalizes the correlation values to be in range [-1,1]:

$$R_{XY}(k) = \frac{\sum_{k=-K}^{K} x(n) * y(n-k)}{\sqrt{e_X + e_Y}} \tag{3.5}$$

where e is the energy of the signal:

$$e = \sum_{k=-K}^{K} s(k)^2 \tag{3.6}$$

With NCCF, a fixed thresholding value can be used when deciding F0 candidates.

3.2.3 Spectrum

Correlation functions operate directly on time series data. An alternative approach is to transfer data to another domain by some mathematical transform operation. A typical use case is using STFT (Short-time Fourier Transform) to carry out a time-frequency decomposition of analyzed signal frames. For each analyzed signal frame, the output of this operation is the magnitude and phase of each frequency component within the signal of interest at a resolution of frame length's proportion to sampling frequency. F0 candidates are obtained by picking the strongest peaks with comb structure from resulting spectrum in desired frequency range.

3.2.4 Cepstrum

Cepstrum is a widely used tool in audio and especially speech processing. It is a time-domain representation although not in a traditional way. It is defined as a inverse Fourier transform of a logarithm of a Fourier transform

$$C = F^{-1}(log(\mid F(x) \mid) \tag{3.7}$$

The resulting cepstrum's units are in the time domain. These *quefrencies* depict the difference of spectral peaks in time. The power of cepstrum is that for a tone with harmonics the first and strongest peak in cepstrum corresponds to F0 in time and it is not sensitive to stronger than fundamental harmonics. Therefore it is very useful for determining the fundamental frequency of human speech or most music instruments.

3.2.5 Post-processing

All of the above mentioned F0 candidate creation methods are providing measures of periodicity from which the F0 candidates are picked. Ideally the peak corresponding to true F0 would be the highest, but often that is not the case. More logic is needed when choosing peaks even though periodicity measurement already is designed to emphasize the most probable peaks. This is called peak-picking and it usually involves some heuristics.

So far we have been considering F0 estimate for a single frame. As we know that vocal fold vibration frequency cannot change abruptly and it usually stays quite stable for longer periods, noisy F0 frame estimates can be further enhanced by taking into account also frames of near vicinity as a post-processing stage. This helps especially in fighting against octave errors, where the estimated F0 is doubled or halved from the real value which can occur often in the raw estimates. This temporal processing is called pitch tracking and it's task is to find the most probable sequence of F0 estimates. Many algorithms, such as RAPT [49], YAAPT [55], and REAPER [50], perform pitch tracking with dynamic programming based on the previous window-level F0 estimates and candidate sets as an input.

Dynamic programming is both optimization method and programming technique. The core principle is to divide problem into smaller sub-problems and instead of recursively solving the whole problem each time, re-use still valid results of previously solved sub-problems.

An example of a pitch track analyzed with 10-ms frame shifts is presented in figure 3.3. The pitch track corresponds to the signal presented in figure 2.3. It is the reference pitch track provided in [42] that has been obtained from an electroglottography (EGG) measurement signal (see section 3.4) processed with the RAPT method presented in Section 3.3.1.

3.3 F0 estimation algorithms

The F0 estimation problem has inspired many researchers to develop different kinds of algorithms to solve the problem. Many of them are based on principles mentioned in 3.2. Here we present prominently used algorithms and take a closer look on the ones chosen as reference methods for the experiments presented in Chapter 5.

3.3.1 RAPT

RAPT (Robust Algorithm for Pitch Tracking) is a famous algorithm by David Talkin [49]. It is based on generalized cross-correlation presented in 3.2.2 as an F0 candidate generator. A major improvement of RAPT from a computational complexity point of view is

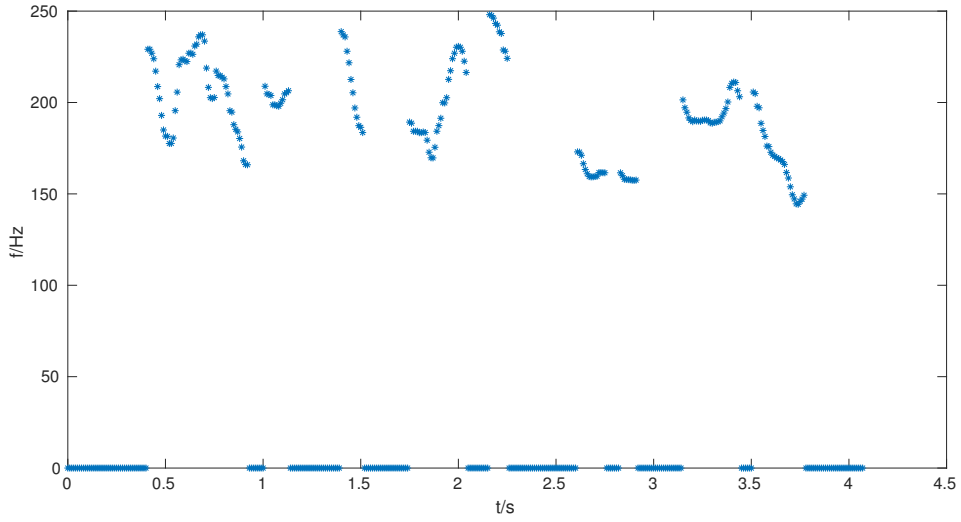


Figure 3.3. *Pitch track of a female spoken utterance "She had your dark suit in greasy wash water all year."*

that correlation is not performed for each candidate frequency but the signal is first down-sampled. From this shorter signal with a coarse time resolution all of the correlation lags required for F0 frequency range are calculated. Candidate lags are collected and further refined with the original signal by cross-correlating in the vicinity of the candidates only. Within the post-processing stage, dynamic programming is used to decide the final F0 estimate or unvoiced decision.

There is no full reference implementation of RAPT openly available, but many speech processing tools or libraries include partial implementations. In the present work, we are using the one implemented within the Speech Signal Processing Toolkit (SPTK) [51]. Only a small portion of the parameters mentioned in the algorithm description [49] are configurable in this implementation.

3.3.2 REAPER

REAPER is a more recent pitch estimation algorithm by David Talkin. More specifically, it estimates the glottal closure instants (GCIs) and F0 estimates are then calculated based on the time differences between two consecutive GCIs. The main tools, NCCF and dynamic programming, are already used in RAPT but, in overall, REAPER is a completely new algorithm. REAPER source code if freely available under Apache license. [50]

As preprocessing, REAPER high-pass filters the signal and possibly performs the Hilbert transform. Then signal features are calculated: GCI candidates are generated by first calculating the LP residual of the signal, then grading the resulting peaks and finally cross-correlating all candidates in the expected pitch range. The voicing decision features in REAPER are based on low frequency energy (normalized by utterance peak energy)

and change in low-passed signal energy, which are used together in determining the voicing status of the frame.

All of the candidate pulses are set in a lattice structure for which dynamic programming is used to calculate probabilities for each pulse period. The best GCI track is obtained by backtracking the most probable path. F0 estimates for requested time stamps are calculated based on GCIs and cross-correlations made in the feature extraction phase.

3.3.3 YAAPT

YAAPT, or Yet Another Algorithm for Pitch Tracking, is a hybrid algorithm working in both time and frequency domains. It was especially developed to perform well with telephony speech alongside high quality recordings. It was originally introduced in [31] and further developed and analyzed in [55].

YAAPT has its roots in RAPT but it adds frequency domain information to enhance robustness. As preprocessing, a new nonlinearly processed version of the original signal is constructed and both signals are then bandpass filtered. Purpose of the nonlinearity is to try to restore possibly missing fundamental period components, which could occur in a high-pass filtered telephony use case. Original YAAPT used the absolute value as the nonlinear function, but latest version opts to the square function. F0 candidate refinement information is obtained by analyzing the spectrogram of the non-linearly processed signal and the normalized low frequency energy ratio is used to enhance the voicing decision.

Frequency domain processing produces F0 candidates with merit values. To reduce pitch halving and doubling, extra candidates may be inserted at the half or double of the highest merit candidate frequency. After combining values as a sequence, a new F0 candidate is added as a 7-point median of the highest merit candidates within the sequence. After dynamic programming, the result is a F0 track with each frame considered to be voiced.

Full accuracy estimates are provided by combining F0 candidates obtained from the spectral track and NCCF from both signals. The final pitch track is obtained by dynamic programming from a matrix containing all of the F0 candidates and corresponding merit values.

3.4 Electroglottography (EGG) or electrolaryngography

Due to the problems mentioned in section 3.1, reliable F0 estimation from speech is a complicated task, and for more accurate results, alternative recording methods can be used. One of the best options would be to actually see the movement of vocal folds, but these methods are intrusive, meaning in practice that endoscope has to be inserted via nasal cavity to proximity of vocal folds. This means that visual inspection of the vocal folds is usually feasible only in a laboratory environment and also makes the operation

both feel uncomfortable and potentially affect the produced speech. Some people cannot tolerate the inserted endoscope and even if they did, in some cases the visibility to glottis can be limited. This and other methods are discussed and experimentally evaluated in e.g. [25].

Less intrusive method is Photoglottography (PGG) where light is emitted through the glottis and the amount of light passing the glottis is measured and mapped into glottis size. Typically this operation also involves the use of endoscope but experiments with completely externally installed light transmitter and sensor have been made e.g. in [48].

A popular very lightly intrusive solution for F0 estimation is the EGG. It is a simple procedure of measuring electrical impedance over the larynx. Electrodes are placed around the subject's neck, and a high frequency but low voltage is applied between the electrodes. The current flowing through the larynx and neighbouring tissues is modulated by the position of the vocal folds in such a way that closed vocal folds result in the highest current (lowest impedance), and open vocal folds in the lowest current (highest impedance). This is due to the fact that vocal fold tissue (with higher surface area while the glottis is closed) has lower impedance than air (glottis open).

The obtained signal can be divided into two components: A high-band component corresponding to the movement of the vocal folds (F0), and a low-band component resulting from other tissue and electrode movement. In F0 estimation, only the high-band is of interest, so high pass filtering for example with a first-order differentiator is applied to remove low frequencies and the DC component [24].

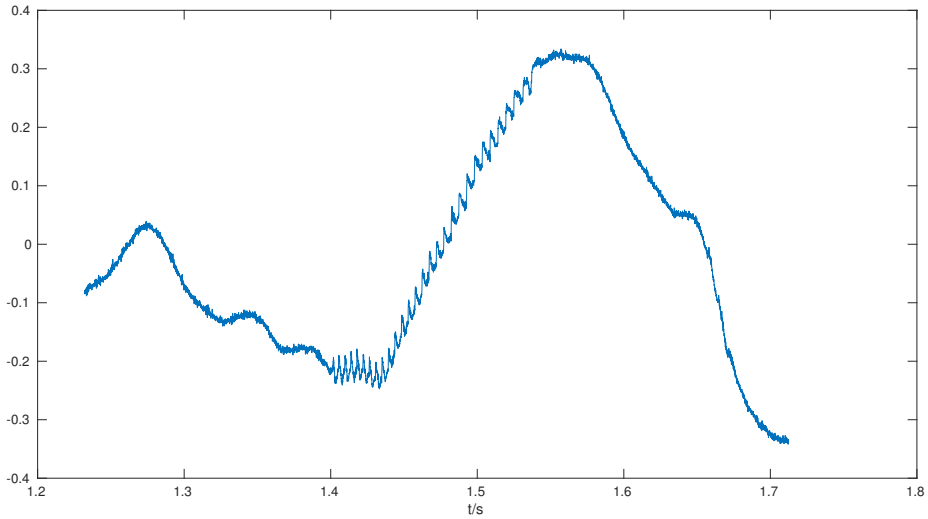


Figure 3.4. *EGG waveform of female spoken word "suit"*

A sample EGG signal is presented in figure 3.4. It is the EGG signal corresponding to the audio signal presented in figure 2.4. The two components mentioned above are clearly distinguishable: The strong slowly changing (low-band) signal from larger muscle movements and a weaker periodic high-band component caused by the vibration of the

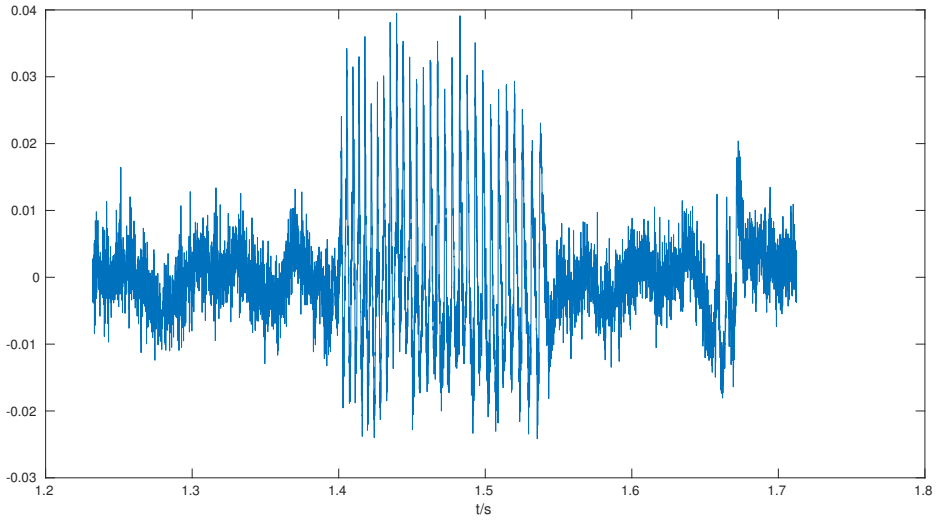


Figure 3.5. *Highpass filtered EGG waveform of female spoken word "suit"*

vocal folds. To obtain the high-band part of interest, this signal needs to be highpass filtered. In figure 3.5 the same signal is filtered with a 2nd order highpass filter having a cut-off at 50 Hz. Now the periodic component between 1.4 s–1.55 s is clearly visible. The filter in this example is not having a linear phase response and a linear phase FIR might be a better option.

When only glottal period is of interest, very good estimation accuracy can be obtained by differentiating the signal and considering the peak values ans GCIs since EGG signal changes abruptly at the moment of glottis opening. However, in some conditions peak can appear as double peak and special care has to be taken to choose the right one. [23]

As EGG is measuring only the overall impedance, it cannot explicitly describe how vocal folds or other parts of the larynx are moving. There is only a limited amount of research done on the topic, such as [20]. What it can do, however, is to measure the timing of the minimum and maximum opening of the vocal cords. This gives highly accurate information of the glottal period corresponding to the F0 of speech.

3.5 Applications of F0 estimation

Reliable F0 estimation is a key part in many speech processing applications. Traditionally all the speech coding, speech analysis and speech synthesis applications have been dependent on separately acquired F0 estimates. Nowadays many speech technology applications are moving towards data driven approaches and especially deep learning where original signal can be the only input parameter and where F0 is not necessarily explicitly represented. F0 estimation is still especially important in speech synthesis, where intonation or other parameters are varied to add naturalness and carry information about feelings etc. Other important application areas are in basic research of speech and

prosody where theoretical background of speech communication is built.

For instance, robust F0 estimate can be used in determining subject's emotional state [43] or health [16] if some history is available. With accurate F0 estimation one possible application is long time analysis of a person's speech parameters. If the F0 (or other properties) are changing over time and health data is available we might be able to notice problems in physical or mental health or even predict problems before they are otherwise observable.

4 NEURAL NETWORKS FOR F0 ESTIMATION

In chapter 3 we discussed traditional F0 estimation algorithms which are based on digital signal processing techniques. That incorporates the use of basic mathematical operations with heuristic models. Machine learning (ML) techniques are able to learn model parameters from the data itself by optimizing a training criterion designed for the particular task. Increasing data amounts and processing power have led to a situation where ML is a viable option to many traditional problems and can often clearly outperform previous approaches.

Artificial neural networks (ANN) are very compelling tools for classification or regression tasks when a large number of labeled data for training the system is available. The main advantage of ANNs over other ML algorithms is that they can learn the statistical properties of raw data such as natural audio or images. With other ML algorithms a user is usually first required to perform dimensionality reduction (i.e., compression) of the data by defining so called features which are descriptors of the data to be fed to the ML system. This approach has clear drawbacks. The user must know the application area and data so well that he can define the essential features. Still, it is very easy to miss some information that is crucial to very high quality results. In theory, when raw data is used as an input to an ANN system, it could learn high-performance features for the given task without domain-specific human expertise.

4.1 Introduction to artificial neural networks

Artificial neural networks are computational systems inspired by the human nervous system. This system facilitates brain-centralized control of the body by having a vast network of interconnected neurons. A biological neuron is a cell that has input and output synapses. If enough input synapses of a neuron activate, the neuron activates its output synapse that acts as an input to other neurons. This process can be interpreted with the model of an *artificial neuron* that is a mathematical data processing model with input and output connections. By the view of this model, the hugely complex network of a human brain performs data fusion of raw messages created by sensory neurons resulting in abstract entities such as feelings or thoughts.

ANNs are typically hierarchical structures which are formed of layers. Each layer uses previous layer's output as an input (or in recurrent case also the same layer's previous

output) which makes neurons within one layer independent of each other forming a parallel system. This property enables efficient implementations of ANNs with modern GPUs and other parallel computing systems.

From a computer science point of view, an ANN can be seen as a computational graph. In mathematics one research area is graph theory. Graphs are networks of nodes, or vertices, connected by the edges. Nodes are entities whose relationships to others are described by the edges. ANNs can be modeled as computational graphs by considering inputs, outputs and mathematical operations as nodes and data flow as edges. Current top ANN libraries TensorFlow [37] and PyTorch [41] work internally with this computational graph principle. This enables the automatic computation of error gradients during backpropagation training for complex network architectures, thereby enabling optimization of model parameters towards a lower error on the training data.

ANNs date back to 1940s and the first AI wave. Back then, both algorithmic and computational restrictions made practical ANNs small and thus unable to solve complex problems. The second rise of ANNs was in 1980–1995 when multilayer network training was made possible by e.g. backpropagation. The third and still ongoing wave started around 2006, and it is characterized by the phrase “deep learning”. This refers to the fact that the amount of training data and available computational power enabled for the first time the functional training of “deep” networks that have more than 2 hidden layers [19].

4.1.1 Artificial neuron and Perceptron

In order to create a neural network, a basic node, neuron, is needed. The most famous neuron model is the Perceptron which actually is an entire ANN but used also as a reference to neuron model [46]. It has an arbitrary number of weighted inputs and a single output. The output is created as a linear combination of weighted inputs and a possible bias term, followed by a non-linearity in the form of binary thresholding. The Perceptron is capable of binary classification and, when having several nodes in parallel, multilabel classification. However, the famous XOR problem, pointed out by Minsky et al. [38], showed that the perceptron was not capable of separating non-linear functions. Therefore Perceptrons in single layers (or two if input and output of a perceptron are both counted as layer) are linear classifiers. The XOR problem along with other criticisms presented by Minsky and Papert [19] was considered so drastic that ANN research was mostly put on hold for a decade and strongly contributed to the forthcoming so called "AI winter". In theory, perceptrons could have been stacked in layers but the used perceptron learning rule did not generalize to multiple layers and therefore training of such network was impossible.

The modern model of the computational neuron is identical to the perceptron in every regard except for the chosen non-linearity: Binary thresholding does not allow error gradients to propagate during backpropagation training, so differentiable functions such as

the hyperbolic tangent, logistic sigmoid, or rectified linear unit are used instead.

4.1.2 Multilayer perceptron

Multilayer perceptron (MLP) or "feedforward net" or "dense net" is a set of interconnected neurons in stages called layers. The first and last layers of a network are called the input and output layers, respectively. The input layer is in the form of an input data vector and the output layer correspondingly produces an output data vector processed by the previous layers of neurons. It has been shown that already one hidden layer in addition to the input and output layers perform the XOR operation and thus solve more complex non-linear problems [47]. To be noted here is that this non-linearity property arises from cascading linear combination operations with a non-linear activation function: Stacking linear units such as perceptrons without non-linear activations still produces linear models.

The real modeling power of ANNs arises when additional “hidden” layers are added between the input and output layer. As these hidden layers have no direct connection to external world, they are called hidden. It has been shown that even with a single hidden layer it is possible to approximate any continuous function, and thus an MLP can act as an universal function approximator. In theory, an infinite number of neurons in a single hidden layer can reach infinite accuracy and complexity of approximation [27]. Implication of this property is that MLP failing in its task must either be poorly trained, have too few hidden units or, in the worst case, is performing a non-deterministic task.

MLP was the second wave of ANN research, made possible especially by backpropagation [47] training algorithm. It was possible to train networks with one or two hidden layers. ANNs still failed to deliver up to expectations and advances in other ML areas led to decrease of research funding and popularity [19].

4.1.3 Convolutional neural networks

Convolutional neural networks (CNNs) have been exceptionally efficient in image recognition tasks. Their main advantage over dense networks is the much smaller number of trainable parameters required for equivalent performance in many tasks, which both makes training of the network easier by making more efficient use of the training data, and the convolutions can be efficiently calculated by modern GPUs. Also, in the case of multidimensional data such as images, CNNs are able to extract local features in spatially correlated data whereas MLPs would ignore the input topology. CNNs were first introduced by LeCun et al. in 1989 [35].

A convolutional layer processes a data tensor consisting of spatially correlated data over a number of channels, and consists of filters (kernels) with which the data is filtered

producing a feature map obtained with parameter sharing from the convolutions, and dense connections over the channels. Filter dimensions and number are fixed. Every filter calculates a feature map which has dimensions depending on both input dimensions and filtering parameters. These feature maps are called channels. In training phase each filter stays constant for each epoch (or batch in case of batch processing) over the data but they are trained between epochs (batches). Filters can be randomly initialized or some suitable pre-trained values can be used to make learning faster. Convolutional layers are trained to recognize features by backpropagation, just like normal fully connected layers in MLPs.

Typical structure of convolution layers is such that the next layer processes somehow downsampled output of the previous layer which results in a hierarchical representation of the input. For example in image classification, the first layer usually finds different kinds of edges, the second layer finds more complex contours and so on, eventually ending in high-level representations such as a car tire. The deepest convolutional layer is often followed by a dense layer in order to carry out a classification decision on the input data.

The aforementioned hierarchy is based on a principle of reducing the data between layers by using each layer as a decimator. The purpose of the decimation is not just to reduce data but also add some translational invariance. The are two main approaches to this. Perhaps the most obvious is the strided convolution in which convolution output is calculated to samples with stride *n*. Stride *n=1* corresponds to calculating convolution to every sample, *n=2* for every other sample and so on. Output will then be a decimation by factor *n*. A more subtle method for downsampling is pooling. When pooling, the convolution is actually performed to every input sample, but the output is formed from a pool of neighbouring convolution outputs. A very common pooling principle is max pooling, in which the output would be the maximum value (i.e., activation) on the pool.

One important aspect in convolutional neural networks is the concept of the receptive field. It is defined to be the image area (or length in time for audio signals) which is taken into account in each convolution operation. For a single layer and normal convolution, it is simply the size of the convolution kernel. However, when stacking layers and reducing data in between, the receptive field would be the union of receptive fields of connections to previous layers. For spatially correlated or time series data, the receptive field is the input area that can affect the output of the given network node.

4.1.4 Recurrent neural networks

Previously introduced networks were feedforward, meaning that information was always flowing forward only. This is enough if the inputs are independent of each other. However, if the there are clear dependencies on previous data points, as in many cases (e.g., time series data such as speech) it is the case, feedforward networks cannot utilize dependencies beyond their receptive fields. Therefore some sort of memory is needed in the

system.

Recurrent neural networks (RNNs) have feedback loops in some of their layers, i.e., they use as input not only the output of the previous layer but also their own previous output. History is then preserved by allowing the previous states of the layer to condition each output. RNNs have shown great performance in e.g. automatic speech recognition where consequent phones and words are clearly dependent.

Recurrent networks were first considered in [47] and each backward connection was shown to be equivalent to an extra layer with the same coefficients, thus making training possible with a similar approach as with feedforward networks with the backpropagation training algorithm.

Theoretically RNNs can have a memory trace of every single input in history. In practice, however, this is not possible and RNNs are difficult to train because of the vanishing gradient problem (see Section 4.1.5). To alleviate the vanishing gradient problem, the Long Short-Term Memory (LSTM) [26] model was introduced. The main contribution was a new "gated" neuron which has an explicit memory state and the ability to update and read from it. The continuous cell state allows for unattenuated gradient propagation through time steps, which greatly enhances the length of the input history that the RNN can utilize.

The LSTM cell has three gates: input, forget and output gate. The name gate comes from the fact that they are controlling, "gating" the data flow. Input gate decides whether to let new data in or not. Forget gate removes the trace of the defined old data. Output gate function is to decide whether the neuron state is affecting the output or not. In addition to the gates, there is also a Cell state variable which gets carried over to be used with the next input.

Different LSTM variants have been very popular with sequential data, including use cases such as speech recognition and natural language processing. A Gated Recurrent Unit (GRU) [14] can be seen as simpler variant of LSTM. GRU cell contains only two gates and no Cell state. Update gate replaces LSTM's Input and Forget gates in deciding how input and previous output is affecting the output. Reset gate determines how much of the previous state is to be forgotten.

Both LSTM and GRU are performing on the same level in general [15], and the winner depends on application and usually both need to be tested. GRU as a simpler unit has the advantage in computational complexity. Vanilla RNNs have been almost entirely replaced by these more advanced recurrent neuron models.

4.1.5 Training of the networks and regularization

It was already mentioned that an ANN is a universal function approximator. This is true on a theoretical level, but in practice it is hard to train perfectly performing networks. Training

refers to the adjustment of network weights on the basis of minimizing prediction error on labeled training data. Here we consider only supervised learning which means the network under training always knows the expected output.

For decades, the dominant method for network training has been stochastic gradient descent (SGD) with error backpropagation made popular by Rumelhart et al. in 1986 [47]. The idea of backpropagation is that input is passed through the network (forward propagation) and prediction error relative to a labeled target is calculated at each layer when propagating the error backwards (backpropagation) in the net. A loss function is needed for error calculation and partial derivatives are calculated for each parameter (weight) of the loss function by using the chain rule of partial derivatives. This gradient points to the direction of increasing error and as we want to minimize the error, we update parameters (weights) to the direction of negative gradient.

A major problem with backpropagation training is the already mentioned vanishing gradient problem: As the error gradient is propagated layer by layer from output to input layer especially through activation functions, the resulting error gradients get smaller and smaller. At some point, the gradient can become so small that either the parameters of the first layers stop updating, or at least the training gets prohibitively slow. As mentioned in Section 4.1.2, for a long time it seemed that only few consecutive layers were possible to train with backpropagation. However, after advancements in many areas (e.g., computational power, deep learning research, amount of data), networks of even more than thousand layers can now be trained with backpropagation [22].

Overfitting and generalization are key concepts of ANN performance. Overfitting relates to ANN (or any ML algorithm) learning the (irrelevant) details of the training data so well that it performs poorly on unseen data due to mismatching details. Generalization is the ability to generalize the mapping function so that only essential features are concerned, thus making prediction perform on a similar level with both previously seen and unseen data.

Regularization is a training tool to reduce overfitting and thus to improve generalization. Various methods of regularization have been proposed. For example, drop-out, weight decay, and batch normalization are widely used regularization methods. These are specific to ANNs, other ML methods usually regularize by the use of penalty term in cost function, which prevents optimization reaching optimum with training data hence creating more generalized model.

In drop-out regularization, some predefined proportion of neurons chosen randomly each training pass are completely neglected. This forces the network to learn the most significant features but also requires more training epochs to converge. This additional training is partly compensated in shorter time for a single epoch since many parameters are zeros. In batch normalization [28], inputs of each layer are normalized to have zero mean and unit variance, usually measured from the minibatch at hand, not the whole training data. Batch normalization especially shortens the needed training time but also works as

a regularization term. Weight decay is a penalty term in the cost function which penalizes large model parameter weight values during training. This basically discourages the model from reaching too specific solutions for the training data, and forces it to use more generic representations.

4.1.6 WaveNet model

One specific type of neural network model architecture is the so-called WaveNet model [40] that was presented by Google DeepMind team for speech synthesis, in which it reached state-of-the-art quality at the time. It was based on their previous work at image generation, namely PixelRNN and PixelCNN models. Its main property is several convolutional layers of dilated convolutions to increase receptive field but limit the number of calculations. Other major building blocks are residual blocks, skip connections and gated convolutions. WaveNet has become very popular in different kinds of audio applications, as it combines a rather large receptive field to a deep network architecture that can handle the vanishing gradient problem. As WaveNet is not recurrent and therefore it also trains quite fast. WaveNet was originally designed for generative purposes, but it was also showed to perform well in discriminatory use cases such as automatic speech recognition.

The foundation of WaveNet is a stack of dilated convolutions. Dilated convolution means that convolution kernel is dilated by adding zeros making the kernel bigger but still having the same non-zero coefficients. When used to increase the receptive field, dilated convolution layers are stacked and greater than one dilation (typically a power of two), is used in the upper layers. Then each layer increases the receptive field by its dilation factor. This allows the CNNs to achieve receptive fields that are functionally comparable to LSTMs, but still are faster to train with GPUs.

In dilated convolution each convolution kernel coefficient is applied to the input signal with a certain integer skip factor, effectively downsampling the input. The difference between a dilated convolution and a strided convolution described in 4.1.3 is that dilation increases the receptive field by downsampling the input while strided only increases the step between consecutive convolutions and thus downsamples the output.

Another key concept of WaveNet is the use of skip connections in order to be able to train such a deep network. Residual skip connections were proposed by He et al. [22] enabling the training of even a thousand-layer network. Skip connections let gradients flow past some layers thus not getting smaller resulting in a vanishing gradient problem. In the present work, we use WaveNet-like architecture for F0 estimation (see Section 4.3).

4.2 Existing ANN based F0 estimators

Articifial neural networks have been used for F0 estimation for decades. Early research, such as [11] were based on hand-crafted features and MLPs. Until the recent rise of deep learning they failed to provide state-of-the-art estimation accuracy. Earlier approaches used ANNs as feature classifiers only while DNN systems mostly operate directly on actual acoustic waveform or spectrogram, thereby giving ANN all the possible information for the task. Some work has still been carried out with classical feature based approach with good results, such as in [21].

It is also possible to mix ANN and DSP methods. One interesting approach to pitch estimation is presented in [32]. ANN, in this case specifically RNN, is used in a regression task of mapping audio signal into single sinusoid presentation. In other words, voiced frames are mapped a into a single sinusoid having a period corresponding to F0 while non-voiced frames are kept as original. Actual F0 is then calculated with autocorrelation. Single sinusoid has a very strong correlation value at lag corresponding to signal period while original signal parts have much lower correlation values. Therefore it is easy to threshold voiced/un-voiced decision and have very accurate F0 estimate from purely sinusoid frames. This method especially differs from most ANN -based methods by making the estimation task to be regression instead of classification. Hence the resolution of the F0 estimate is much higher than with classification based methods. This method was also showed to perform very well with noisy speech.

CREPE [54] is a CNN based method for general pitch estimation. It was developed for music applications but can be used with speech too. CREPE is utilizing a typical CNN structure with six convolutional layers responsible for extracting increasingly complex non-linear signal features and two fully connected layers for input classification into one of 360 different possible frequencies.

CREPE was trained and evaluated with synthesized signals which, on the other hand, makes perfect ground truth signal track, but may also not generalize well enough for real-world recordings. In the original CREPE paper, its performance was measured as state-of-the-art. However later studies such as [10] and [32] report CREPE results with real speech recordings that are inferior to many other methods. CREPE with pre-trained models in different sizes is available as a python module for easy deployment.

4.3 The proposed neural network for F0 estimation

The used ANN model in this work was first presented in [10]. It is inspired by the WaveNet [40] model presented in Section 4.1.6. The formulation of the F0 estimation problem for the proposed ANN model was designed to be interchangeable and comparable to traditional DSP-based methods, which means that the input to the neural network is the raw speech signal that has been split into fixed-length frames at regular intervals. The out-

put of the neural network is a one-hot vector of log(F0) frequency bins within a specified resolution plus the voicing decision.

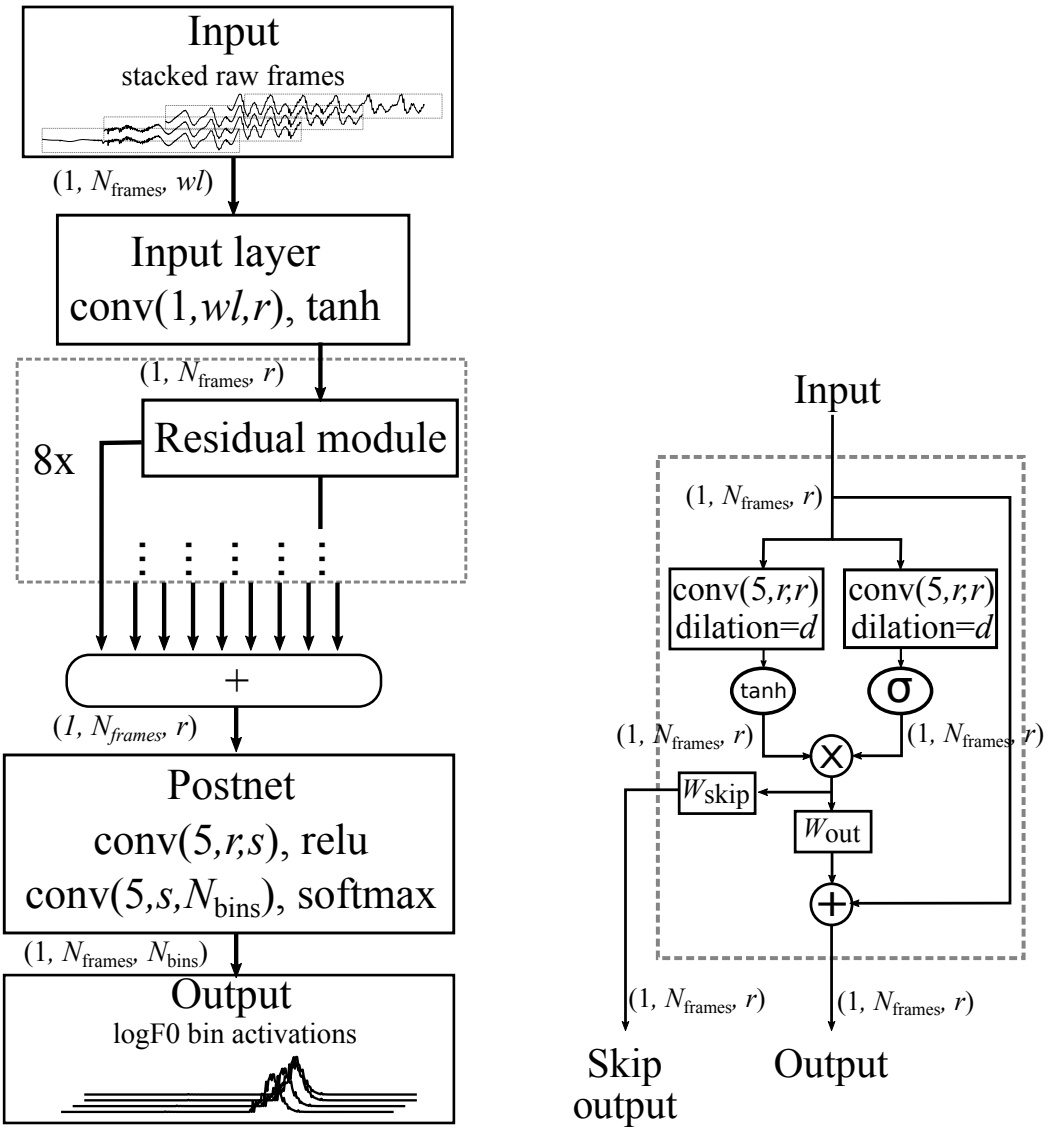
4.3.1 Structure of the proposed neural network

Proposed network is presented in figure 4.1. Input is a tensor of raw signal frames having dimensions $(1, N_{frames}, wl)$, where N_{frames} is the total number of frames and wl the length of each frame. Padding has to be used if original signal is not divisible by wl. This structure is for complying with the traditional problem of F0 estimation of estimating F0 for given speech frame.

The first layer is a convolution with a single coefficient kernel effectively acting as an affine transform to each frame. The purpose of this layer is to remove position dependencies from frames.

The features produced by the input layer are then passed to a residual module stack depicted in detail in figure 4.1b. The first phase in the module is two independent dilated convolutions with different activations. These convolution outputs are then combined and residual is calculated for next round. Also skip connection is provided to the end of the residual module stack where all of the skip connections and the final output connection are summed.

Postnet layer then transforms the features provided by residual module into onehot (only one of the values is non-zero) output vector containing logF0 frequencies and non-voiced decision. The final F0 estimate is obtained by choosing the maximum activation and checking the value. If the value is the unvoiced flag, frame is marked as unvoiced. Otherwise bin number is converted into logF0 value from look-up table and finally converted into Hertz with equation $F0 = e^{(logF0)}$.



***(a)** Overall structure* ***(b)** Residual module*

***Figure 4.1.** Block diagram of the proposed model. Printed with permission from [10]*

5 TEST SETUP

The goal of the present experiments was to investigate the applicability of the WaveNet-inspired F0 estimator presented in Section 4.3.1 to speech in clean speech and noisy telephony speech, and to investigate whether robustness of the network towards telephone speech can be improved through incorporation of coded speech in network training. Simulation environment consisted of signal generation, actual F0 estimation process and result validation and analysis. Everything was run in a Linux environment. Python programming language was used in signal generation and artificial neural network operations. Performance analysis and running of external evaluators were performed with Matlab.

5.1 Test data

If possible, it is a good practice to use completely different test sets for development and validation. Each dataset usually has some unique properties due to recording setup and speaker population. Developing only with a single dataset may lead to suboptimal performance with other datasets. Therefore actual performance evaluation and verification should ideally be done with different dataset(s).

Original signals were containing as clean speech as possible i.e. ambient noise and channel effects were of minimal. To simulate real environment, real ambient noise was added into files and channel effect was simulated by filtering the signal. As the main goal of the research was to develop a robust method for telephony speech, signals were fed through a mobile network simulating system. Simulating properties of mobile telephone was out of the scope of this study but heavy signal processing performed by the handset could have impact on F0 estimation performance.

5.1.1 Data for neural network training

For neural network training, CSTR VCTK corpus [53] was chosen. It is a well-known database of 109 different native English speakers containing approximately 400 sentences uttered by each person. The main drawback in pitch estimation purposes is that this database was intended to automatic speech recognition purposes, thus having no annotated F0 reference. However, high quality of the signals makes reliable automatic pitch

tracking possible, and F0 reference data was generated by the well-performing REAPER algorithm. These estimates were not manually annotated and, as they are calculated from actual speech signal instead of more reliable laryngograph signals, pitch errors are likely. This could be a problem with machine learning techniques since algorithms can learn wrong results. Fortunately, due to the properties of artificial neural networks, this is not seen a major problem since a proper ANN is robust to certain amount of wrongly labeled data.

5.1.2 Data augmentation of training data

Data augmentation for neural network training means generating new synthetic data based on existing one. Augmented data contains the desired information in a different form. Adding noise to audio signal can be considered as data augmentation. It still carries the original information interfered by noise. Purpose of the neural network is to learn the original signal model and neglect non-relevant information.

Noise database used for augmentation on ANN training was NOISEX-92 [52]. It contains e.g. factory and car noise. In addition, a generated random signal (white noise) was used as one of the noise cases. These noise samples were mixed with original speech samples with a predefined signal power and at random start positions of the noise signal. The end result was then filtered with a 17-tap long FIR -filter having random coefficients to simulate different frequency responses and other channel effects. This part of the augmentation process is linear.

Another form of augmentation performed in this study was the use of a speech codec. Speech codecs perform non-linear transformation to speech signal in order to reduce the the amount of data while preserving the information relevant for perceptual signal quality and intelligibility. F0 typically is one of the well preserved properties with only slight variation compared to original uncoded signals, and therefore original labeling can be used as an F0 reference for network training.

The pseudocode presented in Algorithm 5.1 illustrates the functionality of the data augmentation strategy:

5.1.3 Data for F0 estimation evaluation

Generalization is always the key problem with machine learning solutions. ML model might perfectly learn the representation of the training data but perform poorly with other data even though they would be considered as similar. This phenomenon is called overfitting. In extreme cases even within the same data set samples not seen in training phase provide poor results. To fight this overfitting issue with proposed method, completely different data set was used for performance analysis. The one chosen for this purpose was

```
1  if (rand(0,1) > 0.5)
2    sample noise from NOISEX-92
3    sample SNR
4    add noise to original
5  end
6
7  if (rand(0,1) > 0.5)
8    sample 17 FIR coefficients WGN
9    filter signal with FIR
10 end
11
12 if (speechcoding enabled)
13
14   if (AMR-WB enabled)
15     rand=rand(0,1)
16     switch rand
17       case rand < 0.33
18         code signal with AMR
19       case rand > 0.66
20         code signal with AMR_WB
21
22   else
23
24     if (rand(0,1) > 0.5
25       code signal with AMR
```

Algorithm 5.1. *Pseudocode of data augmentation process.*

PTDB-TUG (Pitch Tracking Database from Graz University of Technology) [42]. It contains 4720 sentences extracted from TIMIT corpus [18] read by 10 males and 10 females. In addition to raw audio signals, also laryngograph (described in 3.4) recordings are provided along with example pitch track estimated with RAPT. To be noted here is that this pitch track is automatically generated and not manually checked for estimation errors.

5.2 Simulation environment

Linux PC was chosen to be the platform for the study. It is very well supported by both open source and commercial players. Used tool set was a mixture of open source and commercial programming environments, ready-built executables and self-built executables from open source files.

5.2.1 Python for neural networks

Today Python is the most common environment for ANN development. Probably the main reasons for popularity are ease-of-use and existence of all the major ANN libraries.

Thus Python can be used both quick prototyping without serious programming skills and creating state-of-the-art ANN solutions.

Anaconda Python distribution was used in study. Anaconda contains most of the needed packages by default and conda environment maintenance tool guarantees compatible system by using only verified versions packages and checking their dependencies. For neural network operations TensorFlow backend was used. Proposed method was written in TensorFlow low level API.

5.2.2 Speech processing utilities

So far we have been simulating real-world noise conditions as if we would have just recorded speech in PCM form. In many applications speech transported by telephone network is of interest. Modern networks are not transporting speech signal as such but some encoded version of it. Speech codec can alter the speech signal in many ways thus presenting a big challenge to signal estimators built for clean speech.

Traditional telephone network has approximately 300..3400 Hz frequency range for speech. This is considered as narrowband (NB) speech. Landline networks still operate at this range while mobile networks have recently upgraded to handle wideband (WB) speech too. Typically phone calls are still narrowband quality and AMR codec is used to compress speech to fewer bits. AMR-WB coded with 16kHz sampling frequency (and bandwidth 50..7000Hz) is used in increasing amounts while 4G LTE and beyond networks offer even greater quality speech in form of EVS codec (introduced in [7]) support. In this study we focus on AMR coded speech due to it being the most common case and also the hardest from bandwidth point of view. AMR-WB is also considered as it has been quite widely deployed.

The first step in narrowband network simulation process is filtering the signal with high-pass filter having cut-off at 300 Hz. AMR codec has internal high-pass filtering at 80 Hz as described in [3], but typically devices set their own high-pass filtering in preprocessing phase to match the characteristics of landline network.

5.2.3 Matlab for results and running external utilities

Matlab R2018b was used in creating results figures and running F0 analysis for algoritms except proposed DNN method. YAAPT is fully Matlab code whereas RAPT and REAPER were executables run in Matlab script. Many of the used Matlab scripts were written at Aalto University Acoustics Lab and modified to fit the purpose.

5.2.4 Speech codec

AMR codec top level documentation is presented in [1]. As AMR codec is commercial and licensiable product, only reference code provided by standard [2] is available. From these source codes we built executable encoder and decoder to be run on PC.

Similar licensing issues apply to AMR-WB codec [5, 6] too and same steps were taken to get executable encoder and decoders.

AMR codecs have several possible bit rates that can change dynamically. In this study the bit rate was restricted to be constant (with DTX, Discontinuous Transmission when silent) and the most popular bit rates were chosen: 12.2 kbps for AMR and 12.65 kbps for AMR-WB. 12.2 kbps is the highest possible AMR bit rate and codec acts as GSM-EFR codec [1]. 12.65 kbps is typical bit rate for AMR-WB deployment providing better voice quality with approximately similar bit rate compared to AMR. 12.65 kbps bit rate set higher limit of codec frequency response to 6400 Hz and frequency range from 6400 Hz to 7000 Hz is synthesized based on lower frequency content [5].

As summarized in [13] CELP based speechc codecs such as AMR do change the placing of formants and even F0. On the other hand, in [29] it is stated that F0 is almost unaffected by AMR compression. A change in formant frequency alone can alter the pitch perception but change in actual F0 is a fundamental flaw. Even if we were able to perfectly estimate the F0, coded speech might not contain the real F0 and thus severely limit the possible applications of F0 analysis. Very interesting research topic here is to check whether ANNs can estimate the true F0 from the corrupted speech codec processed speech.

5.3 Reference methods for F0 estimation

We chose as reference methods three established F0 estimation algorithms already presented in 3.3: RAPT, YAAPT, and REAPER. They all are general purpose algorithms to provide good overall performance and also are based on somewhat different basic operations covering most popular techniques. As they are DSP algorithms no training is required and results are easy to reproduce.

RAPT is the simplest one, basing its F0 candidates only on GCC and overall pitch track calculated with dynamic programming. RAPT has been very popular and performed very well with clean speech. The implementation used was the one inside SPTK [51].

YAAPT was interesting because is was designed for use with telephony speech. It tries to restore the filtered out fundamental and, in addition to GCC, also uses frequency domain information.

REAPER was chosen as the third reference, as it represents the latest DSP only generation of F0 tracking algorithms and actually tracks GCIs from LP residuals. F0 estimates

are then calculated from GCI data and dynamic programming is used for actual pitch track calculation.

The most important parameters used in this study are reported in Table 5.1. Sampling frequency was always 16000 Hz corresponding to typical wideband speech. The minimum and maximum F0 estimate frequency were specified in [10] to be 50 and 500 Hz and those same values were supposed to be used with all methods to make comparison fair. Unfortunately in case of YAAPT these parameters were not given and defaults 60 Hz and 400 Hz were used. Frame shift specifies the amount of time difference between consecutive analysis frames. In this study it was chosen to be 10ms. Frame shift may or may not be the same as analysis frame length. The default parameters for frame length were used. SPTK RAPT and REAPER do not specify frame length.

Method parameter list				
Parameter	DNN	RAPT	REAPER	YAAPT
Sample frequency (Hz)	16000	16000	16000	16000
Min Frequency (Hz)	50	50	50	60
Max Frequency (Hz)	500	500	500	400
Frame shift (ms)	10	10	10	10
Frame length (ms)	32	N/A	N/A	35

Table 5.1. *Used parameters in F0 estimation*

5.4 Error metrics

Performance of any estimator is measured by the prediction error of the estimate compared to the true value, "ground truth". In many cases there is no single metric that would provide complete information of the performance. Typical example is a binary decision with heavily skewed distribution where a good estimate would be achieved just by guessing using prior information. Or as another example, not all the errors are as severe and they have to be weighted to get descriptive metric. This leads to use of either very complex single metric or, more often, several simpler metrics with possibly some contradicting properties. From this kind of results educated observer can see what kind of errors estimator usually does and thus take it into account when utilizing the estimator in specific task.

F0 estimation accuracy is a product of two properties: voicing decision and estimated frequency. As we are interested in speech only, we want to estimate the F0 frequency only on voiced speech, not when unvoiced speech or noise is present. This voicing decision is the fundamental property of estimation result, as it marks which parts of the sample are containing speech of interest and it has very noticeable effect on many other error metrics. Typically F0 estimators mark non-voiced frames in resulting frequency vector as 0 or -1 to distinguish them from real F0 frequency estimates.

After voicing decision for the frame is done, the estimated fundamental frequency is reported. This is then compared to reference value and their difference is reported. Frequency error is usually calculated only on frames classified as voiced both reference and F0 estimator under test. Absolute value of error is usually not meaningful metric because in most audio and speech use cases relative error is of interest and in many cases on a logarithmic scale. In this study relative error is presented as error percentage but some studies, especially in the field of music, prefer using *cents* defined as

$$e_{cents} = 1200 * log_2 \frac{x}{y} \tag{5.1}$$

where x is the F0 estimate and y the reference.

5.4.1 Voicing Decision Error

Voicing Decision Error, VDE, is the proportion of incorrectly made voiced/non-voiced decisions.

$$VDE = \frac{N_{incorrect}}{N} * 100 \tag{5.2}$$

where N is the number of frames.

5.4.2 Gross Pitch Error

Gross Pitch Error, GPE, it the proportion of frames correctly classified as voiced but F0 estimate differs more than 20% from the ground truth.

$$GPE = \frac{N_{e>20\%}}{N_{voiced}} * 100\% \tag{5.3}$$

where N is the number of frames.

5.4.3 Fine Pitch Error

Fine Pitch Error, FPE, describes the prediction error in good frames i.e. voicing decision is correct and prediction error is less than 20%. Here error is presented as standard deviation of the relative error.

$$FPE = std(\frac{F0_{ref} - F0_{pred}}{F0_{ref}} * 100\%) \tag{5.4}$$

where $F0_{pred}$ is correctly predicted (voiced frame and GPE < 20%) value vector and $F0_{ref}$ corresponding reference value vector.

5.4.4 F0 frame error

F0 Frame Error, FFE, is the proportion of frames that are either wrongly classified (VDE) or prediction error is more that 20% (GPE). This metric can serve as an overall error if very high accuracy is not needed.

$$FFE = \frac{N_{e>20\%} + N_{voiced}}{N} * 100\% \tag{5.5}$$

where N is the number of frames.

6 RESULTS

This chapter presents the results of F0 estimation experiments. Proposed method is compared to the baseline methods described in 5.3 in different signal conditions. Each method was ran with exactly the same input files containing speech in different noise conditions. Noise was either white or babble noise at four different SNR levels (+15, +5, 0, and -5 dB) in addition to the original clean signals. Input speech to the algorithms was either unprocessed PCM or speech coded with AMR-NB or AMR-WB codecs. The total number of audio files was 4718. In this chapter results are presented in figures and numerical results can be found in Appendix A.

Important thing to notice is that reference pitch tracks (ground truths) were automatically generated with no human supervision so fair amount of errors is expected. This sets the lower limit for error (unless estimators make exactly the same mistakes) and therefore the actual accuracy can be better than the metrics suggest. In this kind of situations one should look at relative errors i.e. how each method compares to others and no absolute numbers.

6.1 Results with non-speech coded signals

Figures 6.1 and 6.2 show the results from the experiments that use test signals without a speech codec. The first main finding is that all compared methods, including all DNN variants with or without AMR coding during training, perform at a comparable level on clean speech. There is a slight advantage for the DNN variants in terms of GPE and FPE, but overall all methods are performing on a good level, similar to [10, 30] . In case of white noise, differences between the methods can be observed already at 5 dB SNR where REAPER and RAPT start to make more voicing decision errors, which is also reflected then in the FFE metric. Around 0 dB, REAPER starts to make a substantial amount of gross prediction errors, while RAPT VDEs increase substantially from 0 to -5 dB. YAAPT seems to perform relatively well down to -5 dB, but is still outperformed by all three DNN variants. For YAAPT it is interesting to note that its baseline performance with clean speech is the worst overall but gets better in high SNR noise conditions.

In case of babble noise, the situation changes somewhat: as the SNR becomes worse, all DSP-based methods (YAAPT, RAPT, REAPER) start to suffer from an increasing number of voicing decision errors and gross errors. YAAPT, a method that is also supposed to be

robust in noisy conditions [55], also suffers from a large number of fine estimation errors (in terms of FPE), and REAPER and RAPT also systematically decease in their fine pitch estimates. REAPER is having major problem with GPE which increases rapidly starting from 5 dB SNR. The DNN variants seem to perform relatively systematically down to 0 dB, but at -5 dB there is already a notable increase especially in VDE (although much smaller than that of the other compared algorithms). YAAPT again shows a bit strange behaviour as VDE suddenly increases along with higher noise levels. This indicates YAAPT confusing speech like signals with speech when speech like noise is close to speech level. RAPT and REAPER are more linear in increase of errors according to the noise level.

As for the use of speech coding as training data preprocessing step, the main finding is that the use of either types of AMR coding have only a very small effect on model performance in clean and noisy signal conditions, which is a desirable property. Overall, the performance of the F0 estimation with the DNN-based variants is at an acceptable level in all SNRs, although performance in severe babble noise is naturally lower than that of white noise.

It is worth noting that FPE actually appears to improve in basically all DNN methods as the SNR degrades in white noise, and also stays at a relatively low level with increasing levels of babble noise for the DNN variants. This somewhat counter-intuitive pattern is primarily caused by the concurrent changes in voicing decisions, where the frames that are still detected as voiced in severe additive noise are actually the ones with very clear harmonic (periodic) structure. In contrast, "difficult" frames are more likely to be discarded from the FPE metrics as the noise increases, since they are more likely to be classified as unvoiced by the algorithms (see also [10]).

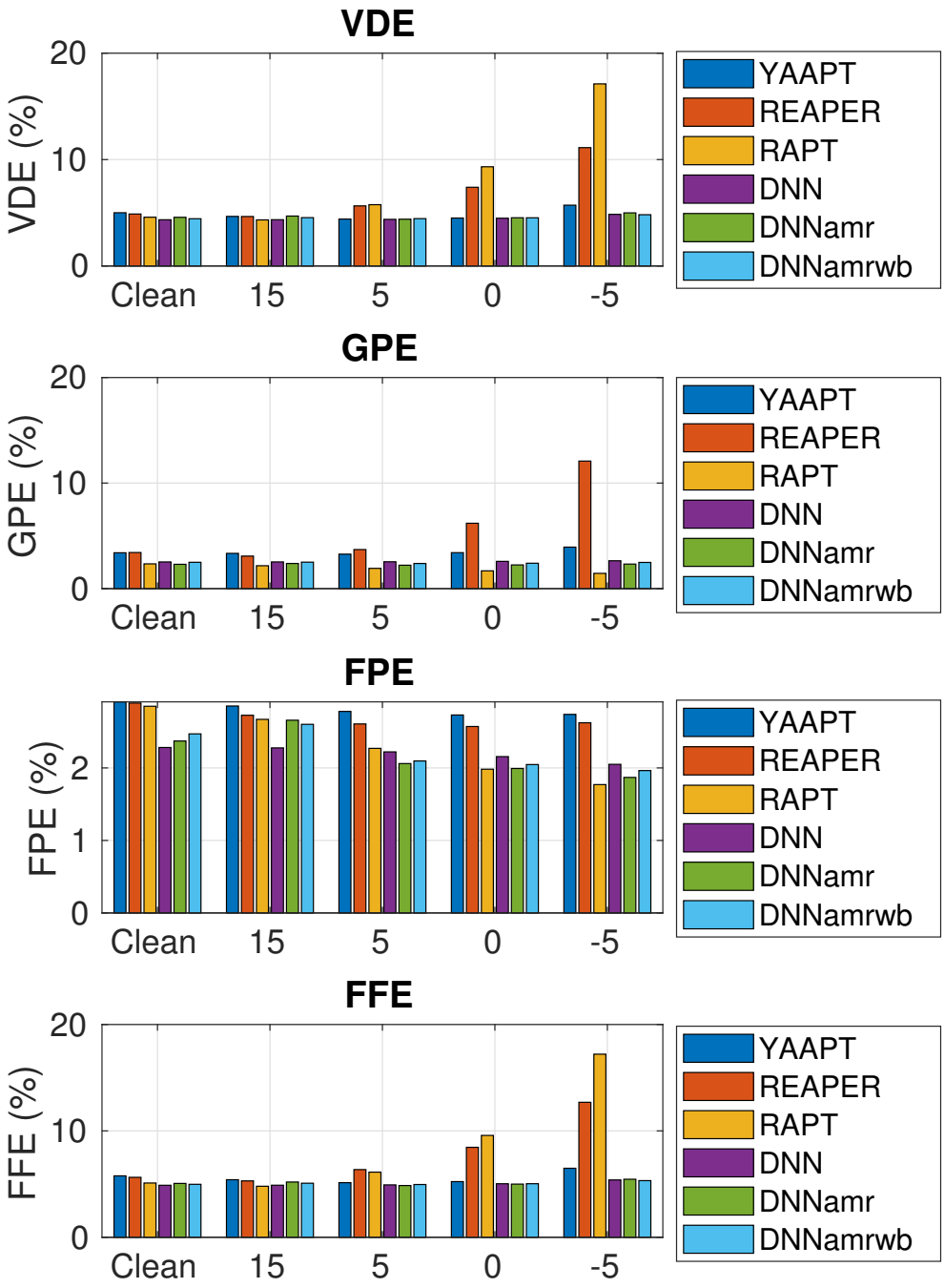


Figure 6.1. *Results of non-coded files with white noise in different SNR scenarios*

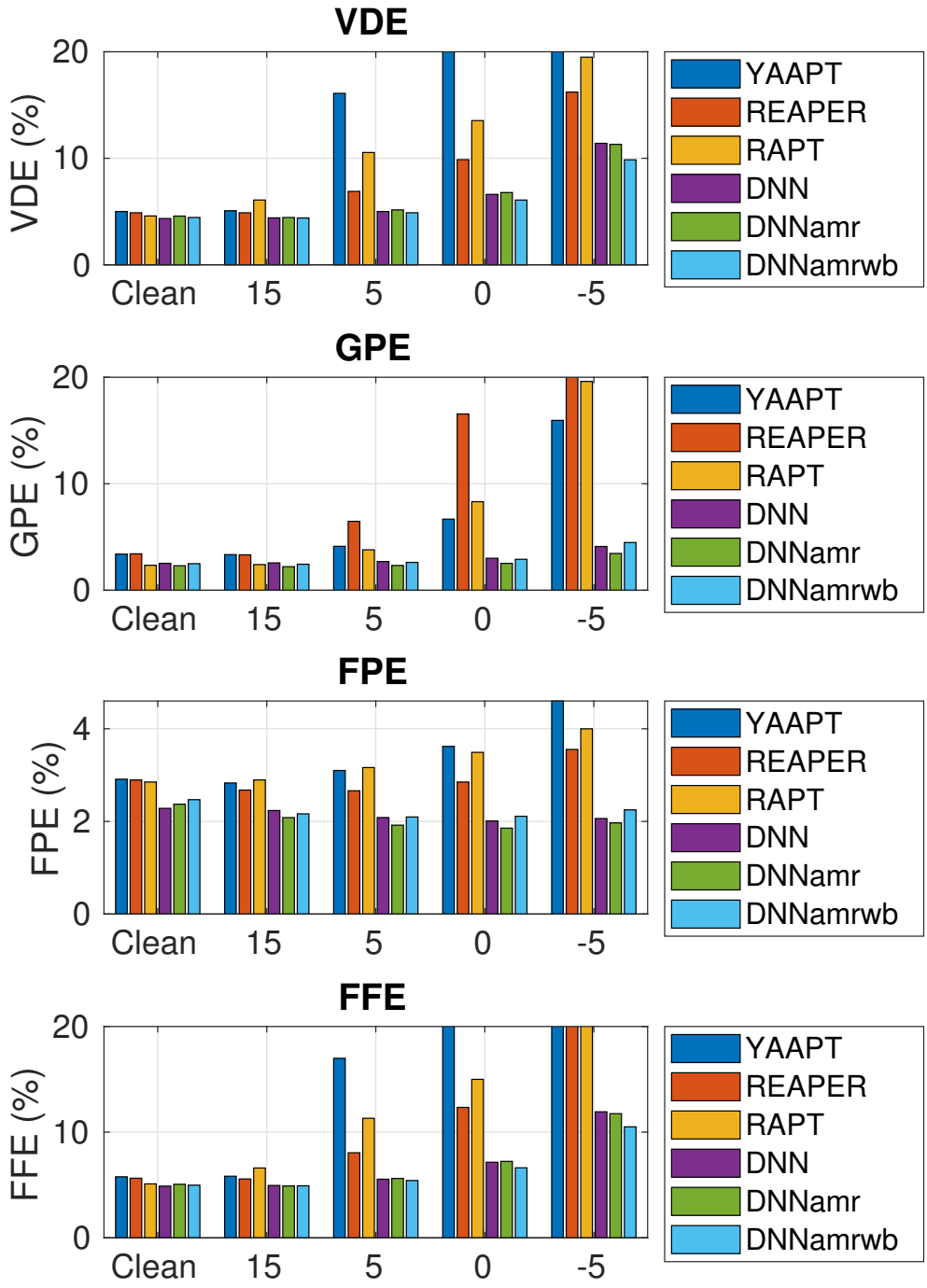


Figure 6.2. *Results of non-coded files with babble noise in different SNR scenarios*

6.2 Results with AMR-NB coded signals

Figures 6.3 and 6.4 show the results from the experiments where each test signal is passed through an AMR-NB codec before the pitch estimation.

One of the most prominent findings is that, in contrast to other methods, REAPER starts to produce a large number of gross errors already at 5 dB SNR with white noise, and even more errors with babble noise. In addition, YAAPT GPE at -5 dB is much worse than without coding of the test signals. Similarly to the non-coded signals, VDEs of all DSP methods start to increase more quickly at low SNRs than those of DNN variants. DNN is having FPE problems with babble noise, for unknown reasons.

Importantly, the inclusion of training samples processed by the speech codecs improves performance from the baseline DNN system, and the difference is most marked in terms of GPE and somewhat less so in FPE. This confirms that the proposed strategy for AMR coding of training data is successful in improving the generalization towards telephone speech, but does not come at a significant cost of reduced performance on non-coded signals (Section 6.1).

One difference in the coded babble noise inputs compared to their non-coded counterparts is that even the DNN variants start to fail in their voicing decisions with more than > 20% VDE, indicating that either the quantization artifacts or the narrowband nature of the coded signal substantially hinders voicing decisions in noisy conditions. Note that no such difference in VDEs is observed when comparing clean signals with or without the codec.

One possible explanation for greatly reduced accuracy in case of AMR -coded babble noise injected files in the use of VAD inside AMR codec as described in [4]. As AMR codec is designed to carry speech as information is may discard other kind of sounds. By the use of VAD codec decides which audio frames should be encoded and which are replaced with comfort noise. In practice this means that frames considered as speech or other kind on sounds of interest are let through and noise frames are discarded. As false negatives (frames incorrectly classified as noise) are very bad, most even remotely speech like parts are coded and transmitted. Encoding probably emphasizes these speech like sounds in the eyes of pitch tracking algorithms and leads to increased false positives (noise considered as speech). Thus, given SNR values are before the AMR codec and true SNR could significantly vary from that.

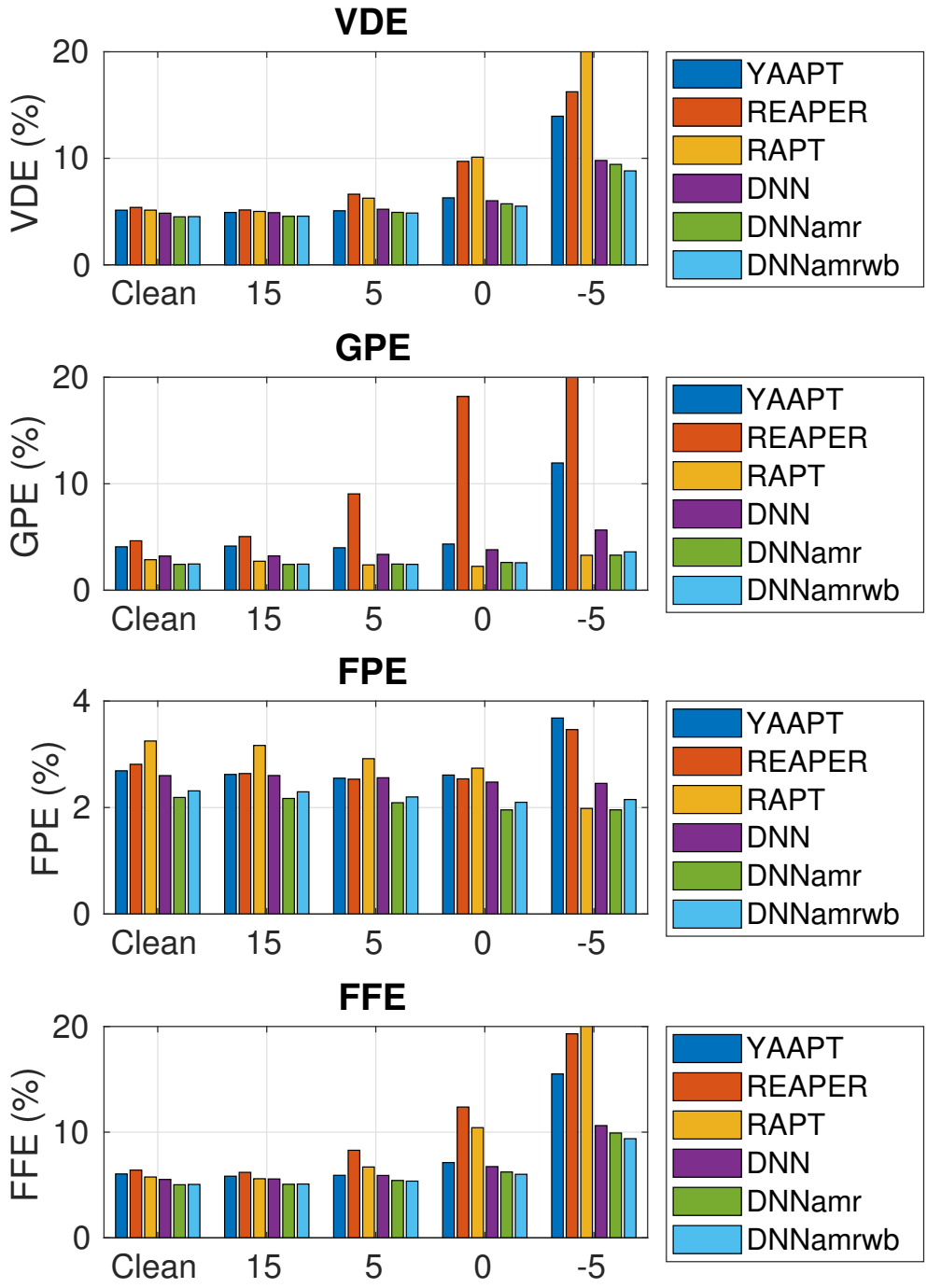


Figure 6.3. *Results of AMR coded files with white noise in different SNR scenarios*

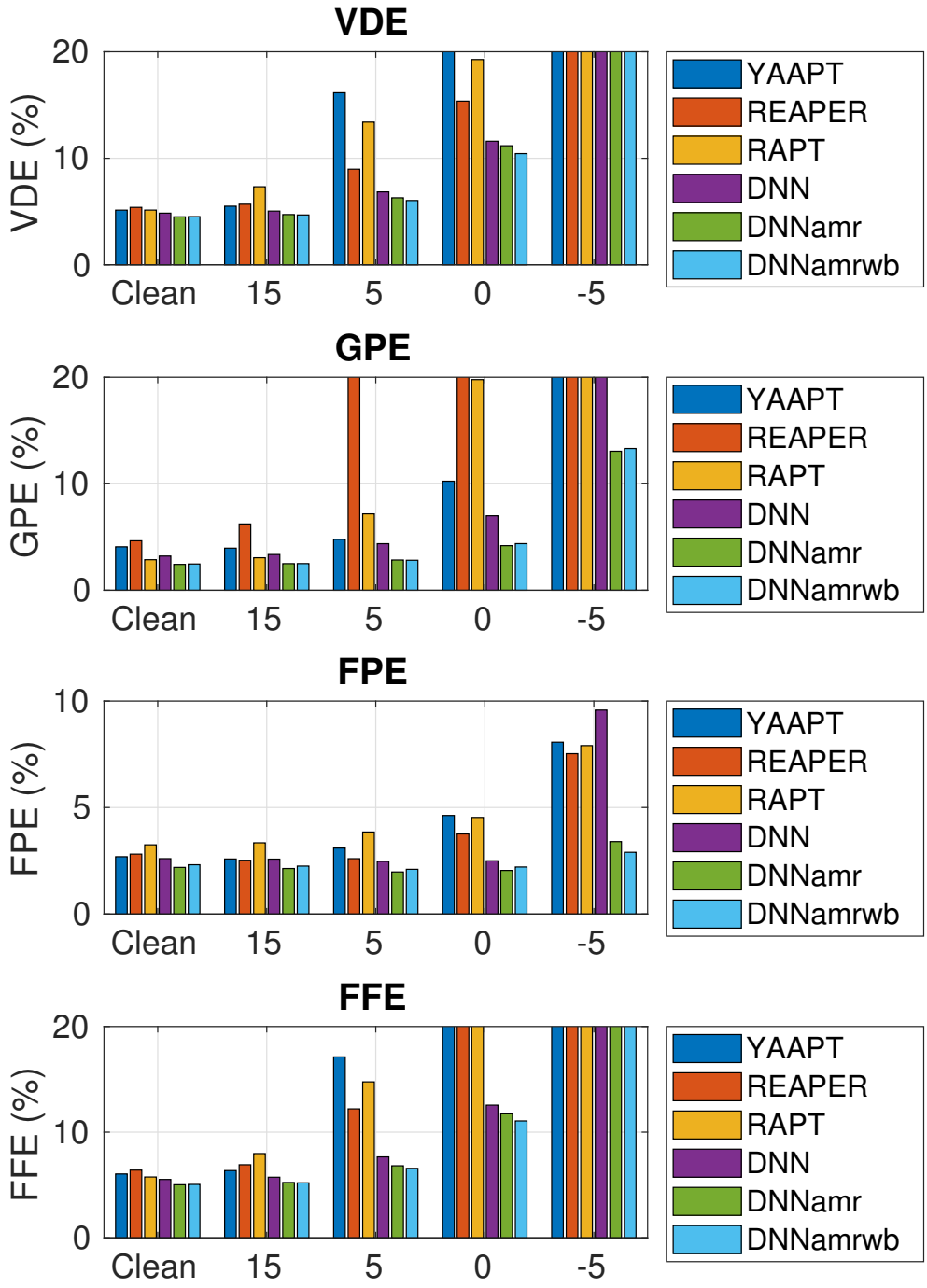


Figure 6.4. *Results of AMR coded files with babble noise in different SNR scenarios*

6.3 Results with AMR-WB coded signals

Finally, figures 6.5 and 6.6 show the experimental results for the case of AMR-WB coded speech inputs.

One might expect the results being somewhere between non-coded and AMR-NB -coded since AMR-WB has frequency range close to non-coded but still has gone through similar processing as AMR-NB -coded signals. Contrary to expectations, AMR-WB coded files seem to in overall provide worse results than AMR-NB. YYAPT is even more suffering from voicing errors especially in babble noise scenario while REAPER improves in GPE point of view.

In general, as with the original and AMR-NB speech, all DNN variants are clearly more accurate than the baseline DSP methods when all metrics are considered.

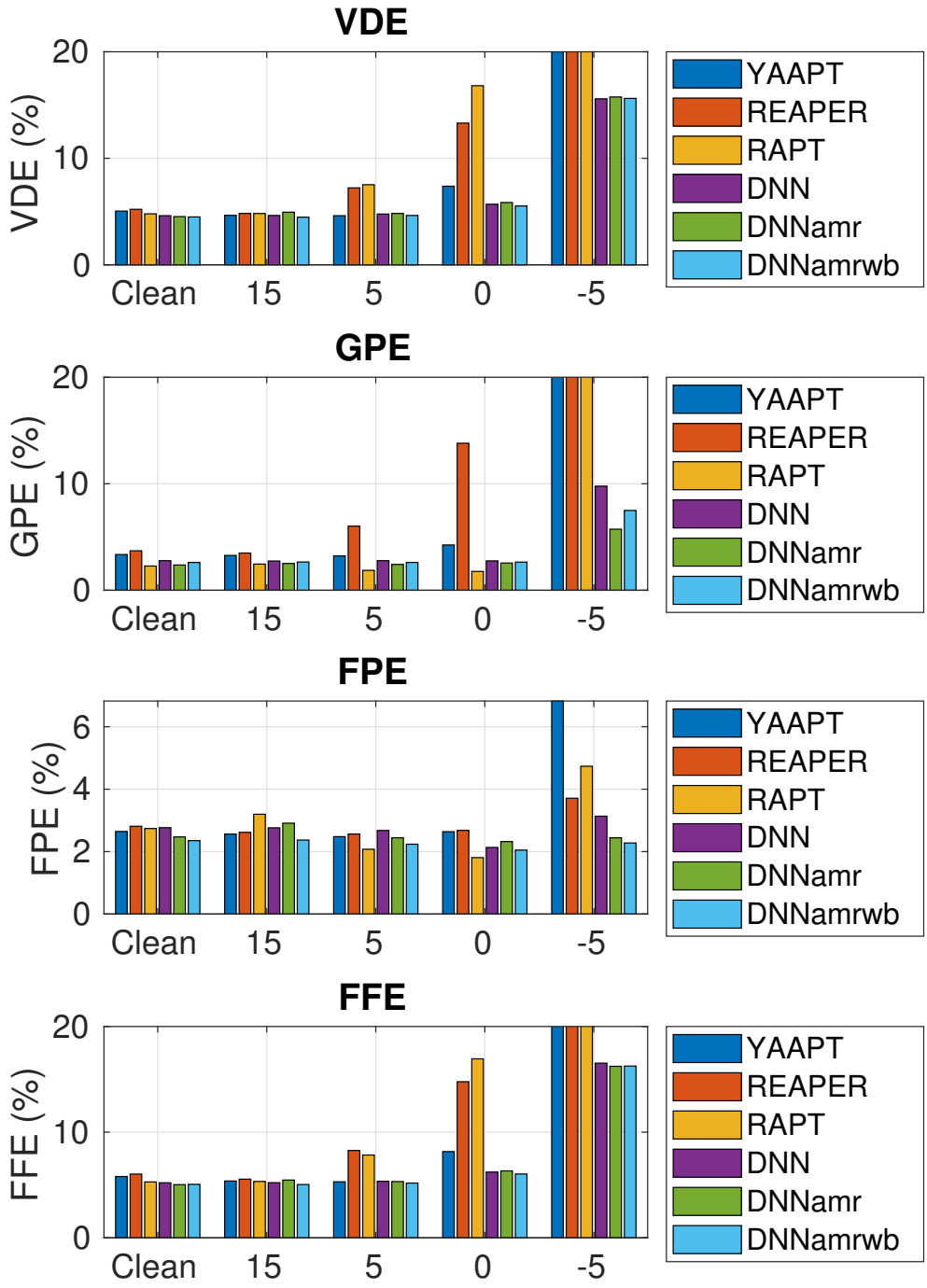


Figure 6.5. *Results of AMR-WB coded files with white noise in different SNR scenarios*

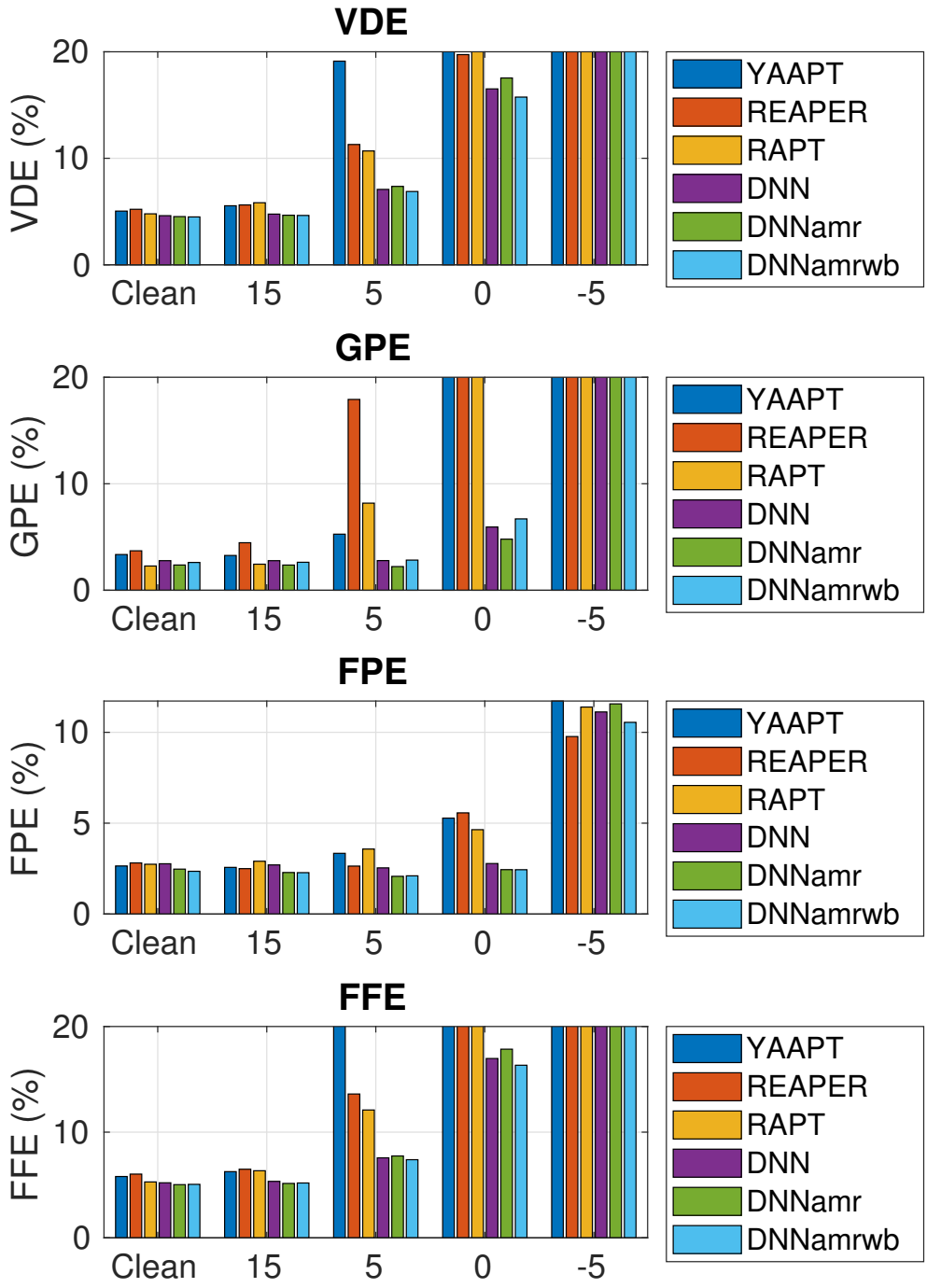


Figure 6.6. *Results of AMR-WB coded files with babble noise in different SNR scenarios*

6.4 Overall impressions

With clean speech all every estimator performed on a relatively similar performance level at ~5%, likely probably setting the lower limit for error. More prominent differences started to be visible at 0 dB SNR white noise and 5 dB SNR babble noise. AMR-NB and AMR-WB codings further increased the error.

YAAPT is designed for telephony speech and indeed AMR-NB coding is not having a big effect on its accuracy. However, for some reason it seems to be the most sensitive to babble noise, making it useless in low SNR babble noise scenarios because of numerous voicing decision errors. White noise performance of YAAPT beats RAPT and REAPER.

RAPT is the most sensitive to white noise and the difference between its white noise and babble noise performance is the smallest in the group. REAPER is in overall performance close to RAPT but, instead of voicing errors, it suffers from severe GPE error. Usually high GPE indicates large amount of octave errors but this was not analyzed in enough detail to make the statement.

DNNs with any tested training procedure seem to be outperforming the other methods consistently. This is especially the case in babble noise while the best case performance is on a same or slightly better level as in the compared methods. The only case where DNN based method had problems was the lowest SNRs with AMR coded speech using the most basic DNN. That is well understandable since the training of DNN was done without AMR coded samples. Still the performance is better than with the DSP based traditional methods.

6.5 Example pitch tracks

To illustrate the real estimation performance on utterance level, example pitch tracks in different conditions are provided in Figures 6.7 and 6.8. The example utterance audio files are mic_F03_si888_amr.wav (6.7) and mic_M06_si1455.wav (6.8) from PTDB-TUG in three forms: clean, 0 dB SNR babble noise and 0 dB babble noise with AMR coding. Clean signal demonstrates the best case performance while 0 dB SNR babble is close to real world worst case. The first utterance (6.7) has been spoken by a female speaker, thus having a relatively high F0, and the second one (6.8) by a male speaker with a considerably lower F0. Performance is demonstrated for two estimators in study, DNNamrwb and RAPT against the EGG signal based reference provided with the dataset.

Figure 6.7a shows the performance with clean and non-coded speech for given female speech sample. The first impression is that both pitch trackers are able to follow reference quite faithfully. RAPT is closer to reference, which was expected since the reference has been calculated with RAPT even though from the EGG signal instead of audio signal. RAPT and reference pitch tracks show very suspicious F0 estimates at 3.9–4.0s and

5.3–5.5s. In those periods pitch estimate is roughly half of the estimates in the rest of the signal, which strongly indicates them being half of the true F0. This is common error with RAPT and, when present in reference, leads to higher than true GPE performance measures even when the evaluated algorithm itself is estimating correctly. DNNamrwb seems to perform better in this sense.

In Figure 6.8a a corresponding comparison is done with male speech sample. In this case both methods seem to perform very well, resulting in nearly identical pitch tracks with the reference. Thing especially to be noted here is that no obvious pitch halvings are present. In case of RAPT this most probably means that half of the true F0 would already be out of range possible F0s and pitch doubling would be more of a concern. That one is not happening either and pitch tracks look very credible. Another thing to note is that, compared to previous female speech example, more severe overshoot is happening at the edges of speech parts. This phenomenon is more severe with DNNamrwb but clearly visible also in RAPT results.

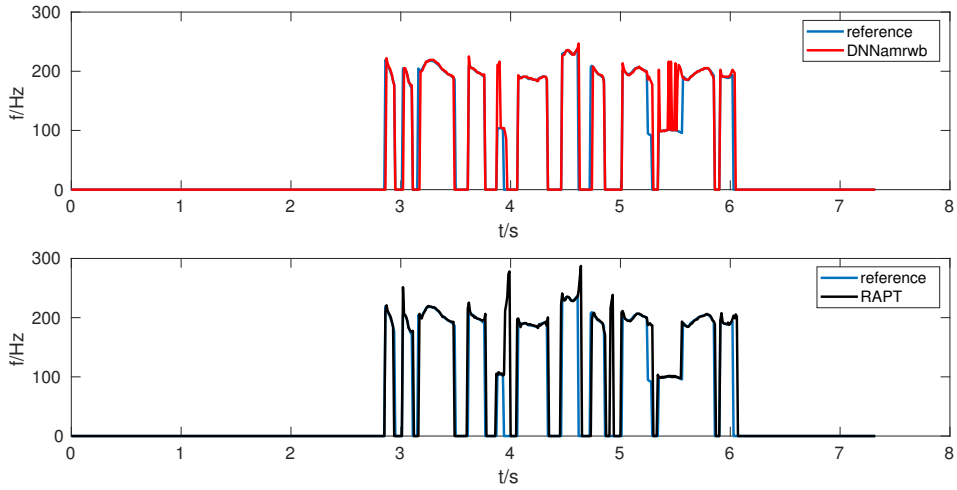
Adding babble noise causes RAPT to make false voicing decisions which is clearly seen in both figures 6.7b and 6.8b. In Figure 6.7b there are three extra consider to be voiced tracks present before the speech starts and in Figure 6.8b two parts after the true speech end. VDE is increased with these false positives. False negatives are absent in this female case (6.7), so for actual speech pitch is tracked still quite well but F0 halving is happening in two occasions. In male speech case (6.8) especially RAPT entirely misses several speechs parts leading to bad VDE score. DNNamr performs close to the clean speech case and clearly outperforms the RAPT.

Compared to babble noise scenario, the AMR codec conditions turns out to be even more challenging. DNNamrwb still performs on a good level in both cases with no false positives but with some false negatives for voicing decisions especially in male speech case. This leads to increased VDE, but in general the result is still good in female speech case while in male speech case false positives are close to RAPT level and estimation quality probably is insufficient for some applications. With female speech (Fig 6.7) RAPT starts to consider even more noise as speech, and the amount of false positives is notable. Some pitch halving is also occurring. In male speech case (Fig 6.8) there is less false positives and pitch halving but the amount of false negatives is considerable. In these cases RAPT clearly is not reliable estimator anymore.

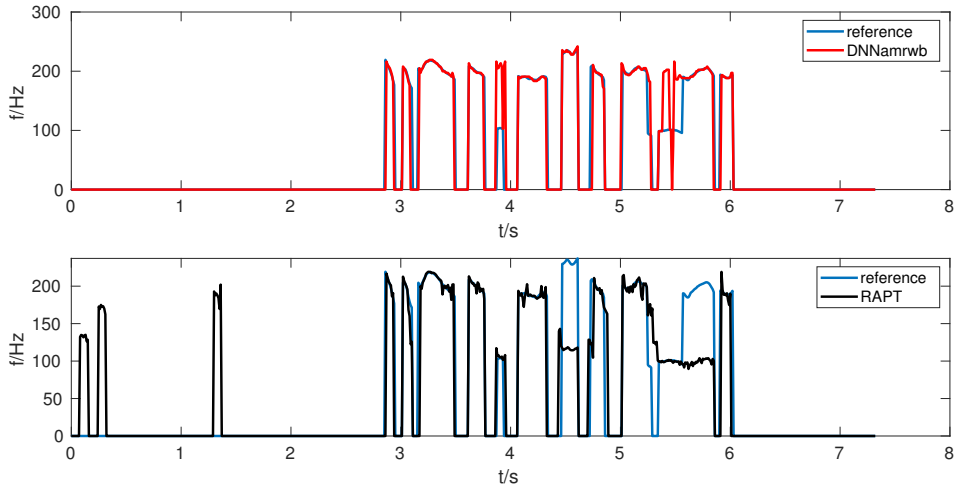
Overall, DNNamrwb performs better than RAPT in these example file cases. DNNamrwb does not make octave errors and is especially robust to false positives in voicing decisions. It seems that DNN trained with speech samples will learn the characteristics of speech and is able to distinguish it from speech-like babble noise. Hence DNN works well as a voice activity detector as well. RAPT

One characteristic of the proposed DNN method seems to be often occuring overshoot in the F0 estimates at the beginning of voiced regions. This behaviour was not prominent in this particular example pitch tracks, but some other signals showed significant error on

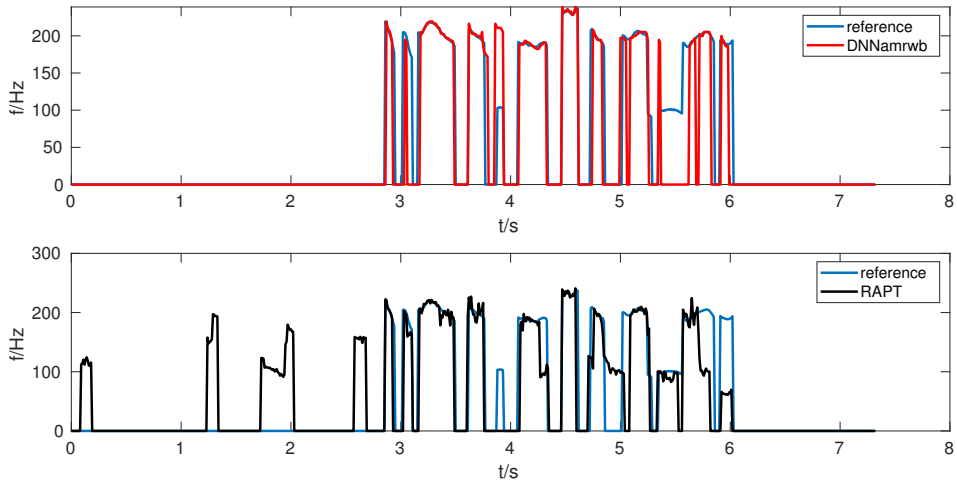
the edges of speech regions. However, as these are of very short duration, the overall average performance is not too much affected. One can also see from the plots, the reference signal has also individual frames with octave errors in the F0, indicating the the ground truth signals are not always unanimous.



(a) *Example pitch tracks with clean speech*

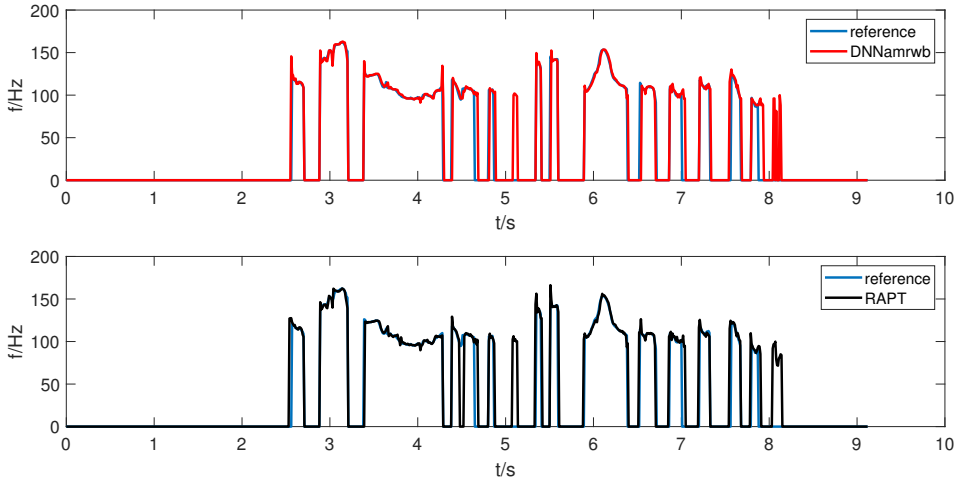


(b) *Example pitch tracks of speech with 0dB SNR babble noise injected*

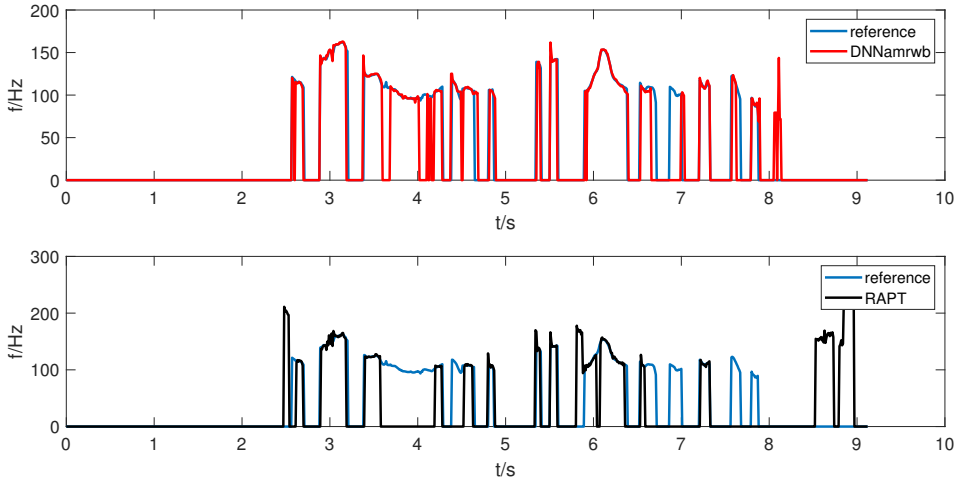


(c) *Example pitch tracks of amr-coded speech with 0 dB SNR babble noise injected*

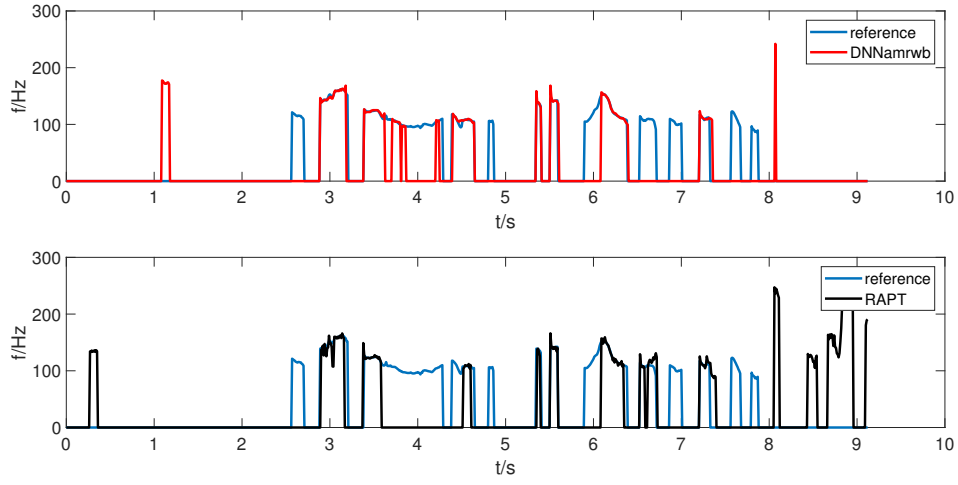
Figure 6.7. *Example pitch tracks for file mic_F03_si888.wav from PTDB-TUG*



(a) *Example pitch tracks with clean speech*



(b) *Example pitch tracks of speech with 0 dB SNR babble noise injected*



(c) *Example pitch tracks of amr-coded speech with 0 dB SNR babble noise injected*

Figure 6.8. *Example pitch tracks for file mic_M06_si1455.wav from PTDB-TUG*

7 CONCLUSION

In this work, a WaveNet inspired CNN introduced in [10] was proposed as a pitch tracking method for telephony speech. Telephony was restricted to mobile networks and AMR codecs, as they are the most prominent medium for speech transmission currently. Especially interesting is the narrowband (landline or AMR coded mobile network) case when the fundamental frequency is basically missing and F0 has to be calculated from other signal properties.

Since the proposed method is ANN based, most crucial parts of the work are model definition and training strategy. The model was kept as in the original paper [10], but training was revised to better take telephony speech into account. This was done by using also AMR and AMR-WB coded samples for model training. Used speech database was CSTR VCTK and F0 ground truths for training were calculated by REAPER algorithm.

The testing of the method and comparison to reference methods was done with a completely different dataset to demostrate the generalization capability of the proposed method. The whole PTDB-TUG database was utilized and speech samples were randomly noise injected and speech coded. These samples were then analyzed with each method and results compared to pitch tracks provided with PTDB-TUG

Results of the study can be summarized in three main points. First, the proposed method outperforms reference method with non-telephony but noise contaminated speech already with very simple training data augmentation and training strategy. Second, data augmentation by speech coding is a valid ANN training strategy also when analyzing non-coded files. Third, AMR -coded speech especially benefits from the use of AMR data augmentation in the training phase.

Future work would be mostly focused on hyperparameter tuning in order to see if performance can still be improved and, on the other hand, if a smaller number of parameters could provide comparable performance. In this work, the original parameter set was not optimized even though the new augmenting methods might have benefited from fine tuning of hyperparameters. For instance, added complexity would allow model to better utilize richer training data if such is available. Smaller model would better scale to real-time device use because of the lesser memory and computational power requirements, perhaps allowing completely new applications of pitch tracking.

Another aspect worth further research is the data augmentation part. An obvious continuation path would include different bit rates of AMR codecs and possibly other codecs

used in speech coding. As this study was conducted only with speech samples in English, some other languages also used in training phase might benefit the generalization capability.

For the method itself, further analysis should be done in order to understand occasional "overshoots" happening at the edges of periodic parts. Otherwise clearly erroneous behaviour was not seen in results.

As a conclusion, this study shows that the proposed method for F0 tracking should be used over existing DSP based methods when strong noise is present in audio signal and/or speech has been processed by telephony network. Absolute error numbers are debatable, since ground truths used for both ANN training and method testing were automatically created by a certain algorithm, thus essentially making comparison to be partly against that algorithm and not a universal truth. However, the relative results are clearly indicating superior performance of the proposed method.

REFERENCES

[1] *3GPP TS 26.071: Mandatory speech CODEC speech processing functions; AMR speech Codec; General description*. Tech. rep. 3rd Generation Partnership Project, June 2018. URL: https://www.etsi.org/deliver/etsi_ts/126000_126099/126071/15.00.00_60/ts_126071v150000p.pdf.

[2] *3GPP TS 26.073 : ANSI-C code for the Adaptive Multi Rate (AMR) speech codec*. Tech. rep. 3rd Generation Partnership Project, June 2018. URL: https://www.etsi.org/deliver/etsi_ts/126000_126099/126073/15.00.00_60/ts_126073v150000p0.zip.

[3] *3GPP TS 26.090 : Mandatory Speech Codec speech processing functions; Adaptive Multi-Rate (AMR) speech codec; Transcoding functions codec*. Tech. rep. 3rd Generation Partnership Project, June 2018. URL: https://www.etsi.org/deliver/etsi_ts/126000_126099/126090/15.00.00_60/ts_126090v150000p.pdf.

[4] *3GPP TS 26.094 : Mandatory speech codec speech processing functions; Adaptive Multi-Rate (AMR) speech codec; Voice Activity Detector (VAD) speech codec; Transcoding functions codec*. Tech. rep. 3rd Generation Partnership Project, June 2018. URL: https://www.etsi.org/deliver/etsi_ts/126000_126099/126094/15.00.00_60/ts_126094v150000p.pdf.

[5] *3GPP TS 26.171: Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; General description*. Tech. rep. 3rd Generation Partnership Project, June 2018. URL: https://www.etsi.org/deliver/etsi_ts/126100_126199/126171/15.00.00_60/ts_126171v150000p.pdf.

[6] *3GPP TS 26.173 : ANSI-C code for the Adaptive Multi-Rate - Wideband (AMR-WB) speech codec*. Tech. rep. 3rd Generation Partnership Project, June 2018. URL: https://www.etsi.org/deliver/etsi_ts/126100_126199/126173/15.00.00_60/ts_126173v150000p0.zip.

[7] *3GPP TS 26.441 : Codec for Enhanced Voice Services (EVS); General overview*. Tech. rep. 3rd Generation Partnership Project, June 2018. URL: https://www.etsi.org/deliver/etsi_ts/126400_126499/126441/15.00.00_60/ts_126441v150000p.pdf.

[8] M. Airaksinen. *Methods for the application of glottal inverse filtering to statistical parametric speech synthesis; Glottaalisen käänteissuodatuksen käyttö tilastollisessa parametrisessa puhesynteesissä*. en. Aalto University publication series DOCTORAL DISSERTATIONS; 109/2018. Aalto University; Aalto-yliopisto, 2018, 102 + app. 84. ISBN: 978-952-60-8028-4 (electronic); 978-952-60-8027-7 (printed). URL: http://urn.fi/URN:ISBN:978-952-60-8028-4.

[9] M. Airaksinen, T. Bäckström, and P. Alku. Automatic estimation of the lip radiation effect in glottal inverse filtering. English. In: *INTERSPEECH, Singapore, Sept.*

14-18, 2014. Interspeech. International Speech Communication Association, 2014, 398–402. ISBN: 978-1-63439-435-2.

[10] M. Airaksinen, L. Juvela, P. Alku, and O. Räsänen. Data Augmentation Strategies for Neural Network F0 Estimation. In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2019. DOI: 10.1109/ICASSP.2019.8683041.

[11] E. Barnard, R. Cole, M. P. Vea, and F. Alleva. Pitch detection with a neural-net classifier. In: *Signal Processing, IEEE Transactions on* 39 (Mar. 1991), 298–307. DOI: 10.1109/78.80812.

[12] P. Boersma and D. Weenink. *Praat: doing phonetics by computer [Computer program]*. URL: www.praat.org.

[13] E. Cheng and I. S. Burnett. On the effect of amr and AMR-WB GSM compression on overlapped speech for forensic analysis. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. 2011, 1872–1875. DOI: 10.1109/ICASSP.2011.5946871. URL: https://doi.org/10.1109/ICASSP.2011.5946871.

[14] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: (June 2014). DOI: 10.3115/v1/D14-1179.

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In: (Dec. 2014).

[16] N. Cummins, A. Baird, and B. W. Schuller. Speech analysis for health: Current state-of-the-art and the increasing impact of deep learning. In: *Methods* 151 (2018). Health Informatics and Translational Data Analytics, 41–54. ISSN: 1046-2023. DOI: https://doi.org/10.1016/j.ymeth.2018.07.007. URL: http://www.sciencedirect.com/science/article/pii/S1046202317303717.

[17] F. Eyben, M. Wöllmer, and B. Schuller. openSMILE – The Munich Versatile and Fast Open-Source Audio Feature Extractor. In: Jan. 2010, 1459–1462. DOI: 10.1145/1873951.1874246.

[18] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. *DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus CDROM*. 1993.

[19] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[20] V. Hampala, M. Garcia, J. G. Švec, R. C. Scherer, and C. T. Herbst. Relationship Between the Electroglottographic Signal and Vocal Fold Contact Area. In: *Journal of Voice* 30.2 (2016), 161–171. ISSN: 0892-1997. DOI: https://doi.org/10.1016/j.jvoice.2015.03.018. URL: http://www.sciencedirect.com/science/article/pii/S0892199715000600.

[21] K. Han and D. Wang. Neural Network Based Pitch Tracking in Very Noisy Speech. In: *Audio, Speech, and Language Processing, IEEE/ACM Transactions on* 22 (Dec. 2014), 2158–2168. DOI: 10.1109/TASLP.2014.2363410.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[23] N. Henrich Bernardoni, C. d'Alessandro, B. Doval, and M. Castellengo. On the use of the derivative of electroglottographic signals for characterization of nonpathological phonation. In: *The Journal of the Acoustical Society of America* 115 (Apr. 2004), 1321–32. DOI: 10.1121/1.1646401.

[24] N. Henrich, C. D'Alessandro, B. Doval, and M. Castellengo. On the use of the derivative of electroglottographic signals for characterization of nonpathological phonation. In: *Journal of the Acoustical Society of America* 115.3 (2004). cited By 141, 1321–1332. DOI: 10.1121/1.1646401. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-1542286741&doi=10.1121%2f1.1646401&partnerID=40&md5=f48fecd793ab393a49021087c2956a1f.

[25] D. P. Hill, A. D. Meyers, and R. C. Scherer. A comparison of four clinical techniques in the analysis of phonation. In: *Journal of Voice* 4.3 (1990), 198–204. ISSN: 0892-1997. DOI: https://doi.org/10.1016/S0892-1997(05)80014-1. URL: http://www.sciencedirect.com/science/article/pii/S0892199705800141.

[26] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. In: *Neural Comput.* 9.8 (Nov. 1997), 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[27] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. In: *Neural Networks* 2.5 (1989), 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: http://www.sciencedirect.com/science/article/pii/0893608089900208.

[28] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, 448–456. URL: http://dl.acm.org/citation.cfm?id=3045118.3045167.

[29] D. Ireland, C. Knuepffer, and S. J. McBride. Adaptive Multi-Rate Compression Effects on Vowel Analysis. In: *Frontiers in Bioengineering and Biotechnology* 3 (2015), 118. ISSN: 2296-4185. DOI: 10.3389/fbioe.2015.00118. URL: https://www.frontiersin.org/article/10.3389/fbioe.2015.00118.

[30] D. Jouvet and Y. Laprie. Performance analysis of several pitch detection algorithms on simulated and real noisy speech data. In: *Proceedings of the 2017 25th European Signal Processing Conference (EUSIPCO), Kos, Greece*. 2017, 1614–1618. DOI: 10.23919/EUSIPCO.2017.8081482.

[31] K. Kasi and S. A. Zahorian. Yet Another Algorithm for Pitch Tracking. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2002, May 13-17 2002, Orlando, Florida, USA*. 2002, 361–364. DOI: 10.1109/ICASSP.2002.5743729. URL: https://doi.org/10.1109/ICASSP.2002.5743729.

[32] A. Kato and T. Kinnunen. Waveform to Single Sinusoid Regression to Estimate the F0 Contour from Noisy Speech Using Recurrent Deep Neural Networks. In: *Proc. Interspeech 2018*. 2018, 327–331. DOI: 10.21437/Interspeech.2018-1671. URL: http://dx.doi.org/10.21437/Interspeech.2018-1671.

[33] P. Keating and G. Kuo. Comparison of speaking fundamental frequency in English and Mandarin. In: *Journal of the Acoustical Society of America* 132 (2012), 1050. DOI: 10.1121/1.4730893.

[34] U. Laine. Modelling of LIP radiation impedance in Z-domain. In: (Jan. 1982). DOI: 10.1109/ICASSP.1982.1171841.

[35] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. In: *Neural Comput.* 1.4 (Dec. 1989), 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541. URL: http://dx.doi.org/10.1162/neco.1989.1.4.541.

[36] L. Lehto, L. Laaksonen, E. Vilkman, and P. Alku. Changes in Objective Acoustic Measurements and Subjective Voice Complaints in Call Center Customer-Service Advisors During One Working Day. In: *Journal of Voice* 22.2 (2008), 164–177. ISSN: 0892-1997. DOI: https://doi.org/10.1016/j.jvoice.2006.08.010. URL: http://www.sciencedirect.com/science/article/pii/S0892199706001135.

[37] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[38] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[39] S. MR and B. Atal. “Code-excited Linear Prediction (CELP): High Quality Speech at Very Low Bit Rates,” in: vol. 10. May 1985, 937–940. DOI: 10.1109/ICASSP.1985.1168147.

[40] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In: *Arxiv*. 2016. URL: https://arxiv.org/abs/1609.03499.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in PyTorch. In: *NIPS Autodiff Workshop*. 2017.

[42] G. Pirker, M. Wohlmayr, S. Petrik, and F. Pernkopf. A Pitch Tracking Corpus with Evaluation on Multipitch Tracking Scenario. In: *Interspeech* (2011), 1509–1512. URL: https://www.spsc.tugraz.at/tools/ptdb-tug.

[43] L. Probst and A. Braun. The effects of emotional state on fundamental frequency. In: *Sasha Calhoun, Paola Escudero, Marija Tabain and Paul Warren (eds.) Proceedings of the 19th International Congress of Phonetic Sciences, Melbourne, Australia 2019*.

[44] V. Pulkki. and M. Karjalainen. *Communication Acoustics: An Introduction to Speech, Audio and Psychoacoustics*. Wiley, 2015. ISBN: 9781118866542.

[45] L. R. Rabiner and R. W. Schafer. Introduction to Digital Speech Processing. In: *Found. Trends Signal Process.* 1.1 (Jan. 2007), 1–194. ISSN: 1932-8346. DOI: 10.1561/2000000001. URL: http://dx.doi.org/10.1561/2000000001.

[46] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. In: *Psychological Review* (1958), 65–386.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-propagating Errors. In: *Nature* 323.6088 (1986), 533–536. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

[48] E. Suthau, P. Birkholz, A. Mainka, and A. P. Simpson. Non-invasive photoglottography for use in the lab and the field. In: (Oct. 2016), 1–5. ISSN: null.

[49] D. Talkin. A robust algorithm for pitch tracking (RAPT). In: *Speech Coding and Synthesis*. Ed. by W. B. Kleijn and K. K. Paliwal. Elsevier Science B.V., 1995, 497–518.

[50] D. Talkin. *REAPER: Robust Epoch And Pitch EstimatoR*. 2014. URL: https://github.com/google/REAPER (visited on 04/02/2019).

[51] K. Tokuda, K. Oura, et al. *Speech Signal Processing Toolkit (SPTK)*. URL: http://sp-tk.sourceforge.net/.

[52] A. Varga and H. Steeneken. Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems. In: *Speech Communication* 12 (July 1993), 247–251. DOI: 10.1016/0167-6393(93)90095-3.

[53] C. Veaux, J. Yamagishi, and K. MacDonald. CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit, [sound]. In: (2017). URL: https://doi.org/10.7488/ds/1994.

[54] J. Wook Kim, J. Salamon, P. Li, and J. Pablo Bello. Crepe: A Convolutional Representation for Pitch Estimation. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2018, 161–165. DOI: 10.1109/ICASSP.2018.8461329.

[55] S. A. Zahorian and H. Hu. A spectral/temporal method for robust fundamental frequency tracking. In: *The Journal of the Acoustical Society of America* 123.6 (2008),

4559–4571. DOI: 10.1121/1.2916590. eprint: https://doi.org/10.1121/1.2916590. URL: https://doi.org/10.1121/1.2916590.

A NUMERIC RESULTS BY METHOD

	SNR	Clean	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	5,00	4,66	4,41	4,51	5,71
	AMR	5,13	4,91	5,07	6,29	13,95
	AMR-WB	5,04	4,64	4,61	7,38	29,72
GPE	PCM	3,39	3,34	3,26	3,41	3,93
	AMR	4,08	4,15	3,99	4,34	11,95
	AMR-WB	3,35	3,27	3,23	4,25	58,85
FPE	PCM	2,91	2,85	2,78	2,73	2,74
	AMR	2,69	2,62	2,55	2,61	3,68
	AMR-WB	2,65	2,56	2,48	2,64	6,82
FFE	PCM	5,77	5,41	5,13	5,23	6,49
	AMR	6,04	5,83	5,91	7,12	15,51
	AMR-WB	5,80	5,38	5,30	8,16	34,85

(a) *Clean and white noise*

	SNR	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	5,08	16,10	26,06	33,97
	AMR	5,50	16,15	27,16	39,28
	AMR-WB	5,54	19,12	31,57	38,04
GPE	PCM	3,35	4,13	6,68	15,95
	AMR	3,95	4,79	10,25	43,42
	AMR-WB	3,27	5,27	23,31	68,83
FPE	PCM	2,83	3,10	3,62	4,60
	AMR	2,58	3,10	4,63	8,07
	AMR-WB	2,57	3,34	5,28	11,72
FFE	PCM	5,83	16,99	27,44	36,82
	AMR	6,36	17,12	28,96	44,08
	AMR-WB	6,26	20,18	34,76	42,63

(b) *Babble noise*

Table A.1. *YAAPT results*

	SNR	Clean	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,88	4,65	5,65	7,40	11,12
	AMR	5,39	5,15	6,63	9,71	16,25
	AMR-WB	5,21	4,82	7,22	13,30	22,45
GPE	PCM	3,42	3,08	3,70	6,19	12,08
	AMR	4,64	5,04	9,05	18,20	39,95
	AMR-WB	3,70	3,50	6,02	13,81	43,63
FPE	PCM	2,89	2,73	2,61	2,57	2,62
	AMR	2,81	2,64	2,53	2,54	3,46
	AMR-WB	2,81	2,62	2,56	2,68	3,71
FFE	PCM	5,64	5,30	6,36	8,46	12,69
	AMR	6,41	6,20	8,28	12,38	19,31
	AMR-WB	6,03	5,55	8,27	14,77	22,95

(a) *Clean and white noise*

	SNR	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,88	6,89	9,87	16,21
	AMR	5,68	8,98	15,37	34,37
	AMR-WB	5,62	11,30	19,74	29,54
GPE	PCM	3,33	6,47	16,55	48,56
	AMR	6,23	20,27	48,97	90,68
	AMR-WB	4,46	17,92	66,42	94,53
FPE	PCM	2,68	2,66	2,85	3,55
	AMR	2,52	2,60	3,76	7,53
	AMR-WB	2,50	2,64	5,57	9,77
FFE	PCM	5,57	8,05	12,34	21,88
	AMR	6,91	12,20	21,52	44,30
	AMR-WB	6,49	13,61	23,97	31,88

(b) *Babble noise*

Table A.2. *REAPER results*

	SNR	Clean	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,58	4,33	5,76	9,33	17,11
	AMR	5,13	5,01	6,26	10,11	20,20
	AMR-WB	4,78	4,81	7,52	16,82	23,56
GPE	PCM	2,35	2,17	1,92	1,68	1,44
	AMR	2,87	2,73	2,38	2,25	3,29
	AMR-WB	2,28	2,46	1,87	1,77	79,71
FPE	PCM	2,85	2,67	2,27	1,98	1,77
	AMR	3,25	3,17	2,92	2,74	1,98
	AMR-WB	2,74	3,20	2,08	1,81	4,74
FFE	PCM	5,10	4,80	6,12	9,57	17,21
	AMR	5,76	5,59	6,70	10,42	20,31
	AMR-WB	5,29	5,33	7,84	16,94	23,56

(a) *Clean and white noise*

	SNR	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	6,08	10,55	13,54	19,49
	AMR	7,33	13,40	19,27	29,81
	AMR-WB	5,83	10,69	21,01	29,32
GPE	PCM	2,41	3,80	8,32	19,61
	AMR	3,05	7,17	19,78	50,09
	AMR-WB	2,45	8,18	28,74	68,95
FPE	PCM	2,90	3,16	3,49	4,00
	AMR	3,34	3,85	4,54	7,91
	AMR-WB	2,91	3,58	4,64	11,39
FFE	PCM	6,61	11,31	15,00	21,77
	AMR	7,97	14,76	22,12	32,83
	AMR-WB	6,35	12,10	23,15	31,11

(b) *Babble noise*

Table A.3. *RAPT results*

	SNR	Clean	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,35	4,35	4,38	4,49	4,85
	AMR	4,85	4,90	5,22	6,01	9,80
	AMR-WB	4,61	4,63	4,77	5,69	15,59
GPE	PCM	2,53	2,53	2,54	2,58	2,64
	AMR	3,21	3,23	3,37	3,81	5,66
	AMR-WB	2,78	2,75	2,78	2,76	9,78
FPE	PCM	2,28	2,28	2,22	2,16	2,05
	AMR	2,60	2,60	2,56	2,48	2,45
	AMR-WB	2,77	2,76	2,68	2,13	3,13
FFE	PCM	4,89	4,90	4,93	5,03	5,39
	AMR	5,52	5,57	5,90	6,74	10,62
	AMR-WB	5,21	5,22	5,35	6,22	16,53

(a) *Clean and white noise*

	SNR	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,41	5,00	6,61	11,40
	AMR	5,04	6,84	11,60	24,41
	AMR-WB	4,76	7,08	16,51	24,12
GPE	PCM	2,57	2,70	3,00	4,11
	AMR	3,36	4,37	6,99	45,11
	AMR-WB	2,78	2,79	5,94	65,96
FPE	PCM	2,23	2,08	2,01	2,06
	AMR	2,58	2,47	2,50	9,58
	AMR-WB	2,70	2,54	2,78	11,12
FFE	PCM	4,95	5,55	7,15	11,93
	AMR	5,73	7,66	12,56	25,43
	AMR-WB	5,34	7,57	16,97	24,28

(b) *Babble noise*

Table A.4. *DNN results*

	SNR	Clean	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,57	4,69	4,40	4,53	4,99
	AMR	4,51	4,56	4,93	5,73	9,43
	AMR-WB	4,53	4,94	4,82	5,85	15,76
GPE	PCM	2,30	2,38	2,22	2,24	2,32
	AMR	2,43	2,43	2,46	2,61	3,30
	AMR-WB	2,37	2,52	2,43	2,56	5,73
FPE	PCM	2,37	2,66	2,06	1,99	1,87
	AMR	2,19	2,17	2,09	1,96	1,96
	AMR-WB	2,47	2,91	2,44	2,32	2,44
FFE	PCM	5,06	5,20	4,87	5,00	5,46
	AMR	5,02	5,07	5,43	6,23	9,92
	AMR-WB	5,03	5,47	5,32	6,33	16,23

(a) *Clean and white noise*

	SNR	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,44	5,16	6,79	11,31
	AMR	4,72	6,29	11,18	21,41
	AMR-WB	4,65	7,36	17,54	23,91
GPE	PCM	2,22	2,33	2,53	3,46
	AMR	2,50	2,84	4,19	13,05
	AMR-WB	2,37	2,22	4,80	67,06
FPE	PCM	2,08	1,92	1,85	1,97
	AMR	2,14	1,98	2,05	3,40
	AMR-WB	2,28	2,08	2,44	11,56
FFE	PCM	4,91	5,62	7,23	11,75
	AMR	5,24	6,82	11,73	21,77
	AMR-WB	5,15	7,75	17,85	24,01

(b) *Babble noise*

Table A.5. *DNN$_{AMR}$ results*

	SNR	Clean	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,44	4,54	4,45	4,53	4,82
	AMR	4,52	4,57	4,86	5,50	8,82
	AMR-WB	4,49	4,47	4,63	5,52	15,62
GPE	PCM	2,50	2,51	2,38	2,41	2,48
	AMR	2,47	2,45	2,43	2,59	3,61
	AMR-WB	2,61	2,65	2,61	2,64	7,49
FPE	PCM	2,47	2,60	2,10	2,05	1,96
	AMR	2,31	2,30	2,20	2,10	2,15
	AMR-WB	2,35	2,37	2,24	2,05	2,28
FFE	PCM	4,98	5,08	4,96	5,04	5,33
	AMR	5,05	5,09	5,36	6,01	9,38
	AMR-WB	5,06	5,04	5,18	6,04	16,26

(a) *Clean and white noise*

	SNR	15 dB	5 dB	0 dB	-5 dB
VDE	PCM	4,39	4,89	6,08	9,86
	AMR	4,68	6,04	10,44	20,91
	AMR-WB	4,63	6,88	15,75	23,87
GPE	PCM	2,44	2,62	2,91	4,48
	AMR	2,50	2,82	4,38	13,31
	AMR-WB	2,63	2,83	6,70	61,34
FPE	PCM	2,16	2,10	2,11	2,25
	AMR	2,25	2,10	2,21	2,90
	AMR-WB	2,28	2,10	2,44	10,55
FFE	PCM	4,92	5,42	6,62	10,51
	AMR	5,21	6,57	11,06	21,33
	AMR-WB	5,19	7,39	16,33	23,95

(b) *Babble noise*

Table A.6. *DNN$_{AMR-WB}$ results*