University ”Politehnica” of Bucharest
Faculty of Electronics, Telecommunications and Information Technology

Deep Learning Techniques for Vocal Pathology Detection

Diploma Project

Submitted in partial fulfillment of the requirements
for the degree of *Engineer*
in the domain of *Electronics, Telecommunications and Information Technology*
study program *Information Engineering*

Thesis Advisor(s)
Prof. PhD. Eng. Corneliu Burileanu
Asst. Prof. PhD(c). Eng. Ana-Antonia Neacșu

Student
Robert-Valentin Bencze

Year 2022

Table of Contents

List of figures

List of tables

List of abbreviations

ADC = Analog to Digital Converter
API(s) = Application Programming Interface(s)
AUC = Area Under Curve
CCE = Categorical Crossentropy
DFT = Discrete Fourier Transform
DTFT = Discrete Time Fourier Transform
EGG = Electroglottograph
ELU = Exponential Linear Unit
EMG = Electromyographical/Electromyography/Electromyograph
FCNN = Fully Connected Neural Network
FNR = False Negative Rate
FPR = False Positive Rate
IDFT = Inverse Discrete Fourier Transform
IDTFT = Inverse Discrete Time Fourier Transform
MFCC(s) = Mel-frequency cepstral coefficient(s)
PMF(s) = Probability Mass Function(s)
PSNR = Peak Signal-to-Noise Ratio
ROC = Receiver Operating Characteristic
ReLU = Rectified Linear Unit
SMAD = Squared Median Absolute Deviation
STFT = Short Time Fourier Transform
SVD = Saarbrucken Voice Dataset
ZCR = The Zero-Crossing Rate

Introduction

The most recent advancements in Deep Learning have driven massive changes in the healthcare area, with applications ranging from the protein folding problem [3] to diagnosis assistants [4]. The COVID-19 pandemic has emphasized the benefits of remote activities including pathology diagnosis, hence an automatic pathology diagnosis assistant may prove useful to help medical professionals in the pathology assessment procedures even outside the context of a pandemic. Voice pathology detection is the study of diseases using the quality of voice as an indicator. Voice alterations usually provide early indications of different diseases, such as Parkinson's, because voice quality can be affected by multiple issues, such as: irritation, neurological or muscular issues and a variety of other factors that ultimately influence the vocal signal [5]. Most of the research in automatic vocal pathology detection has been conducted on binary classification problems with significant progress yet to be achieved in the multi-class problem.

Related work in the field approaches the classification task using feature representations with focus on various characteristics. The authors of VoiceLens [6] proposed a smart health monitoring system capable of detecting multiple neurological disorders based on vocal signal. The authors achieved accuracies between 60.3% and 94.3% with deterministic and deep learning based features, extracted from audio recordings of healthy and pathological voices from datasets containing 3 to 7 classes. The minimum accuracy for 7 classes was 60.3%, while the maximum accuracy for 3 classes was 94.3%. The extracted features have been classified with various architectures, such as Deep Neural Networks, Support Vector Machines and other types of classifiers. However, the authors did not specify how the hand-crafted features were extracted from the input signals (i.e. if the features were extracted from windows of each input signal or from the entire duration of the signal) or if any compression was applied to the features prior to classification.

In [7], the authors achieved a maximum accuracy of 94.6% and 99.42% for two different datasets containing vocal recordings labeled with 3 classes. The paper presents a frame-based classification method, where multiple features are extracted from the frames of audio signals and are classified by Support Vector Machines or K-Nearest Neighbours.

The authors of [8] propose an Artificial Neural Network as a multi-class discriminator uses compound features extracted from audio and electromyographical (EMG) recordings of the subjects' speech. The proposed system has achieved a 94.4% accuracy on the audio signals' feature representation and a 71% accuracy on the EMG dataset. However, in this approach, an implementation of a combined dataset of EMG and audio signals was not considered.

The authors of [9], [10], [11], [12] and [13] have all proposed binary classifiers for pathology detection, where the best achieved accuracies were 94.6%, 84.37%, 91.17%, 95% and 87.11% respectively. Their methods used various classifiers, such as: Support Vector Machines, Convolutional Neural Networks, Online Sequential Extreme Learning Machines or Recurrent Neural Networks on different feature representations of the vocal signals.

However, neither method specified how the vocal recordings were split if each subject was recorded multiple times in the dataset, nor training graphs were provided for the evolution of the loss functions of the classifier models.

The thesis proposes an approach that achieves 95% mean accuracy based on a convolutional neural network to solve a multi-class pathology classification problem, using Mel-Frequency Cepstral Coefficients extracted from a combined dataset of electroglottographic and audio recordings of human speech.

Motivation

The motivation for implementing the proposed system originates in the need of remote diagnosis for affections deemed as non-urgent during lockdown periods, doubled by the reduced volume of research in the multi-class approach of vocal pathology discrimination based on the combination of data acquired by multi-modal signal recording methods. Additional motivation for the choice of the project include:

- The preference of non-invasive diagnosing methods over invasive methods due to the increased safety of non-invasive methods
- The cost-effectiveness of Deep Learning based diagnostication methods
- The need of faster and more accurate diagnostics

Objectives

The following work proposes a system that aims to:

- study a multi-modal dataset consisting of EGG and vocal recordings of human speech
- explore an approach to the problem of vocal pathology diagnosis that combines data acquired by multiple methods
- implement a modular and highly flexible algorithm that can be used for various audio classification tasks
- extract feature representations from the signals of the multi-modal dataset
- train multiple neural networks for each classification approach: EGG, vocal and combined dataset
- achieve a mean accuracy above the average 71.4% accuracy of medical professionals [14]

Applicability

The applicability of such a flexible audio classification system ranges from the medical domain, where its uses can aid the accuracy of diagnostics and decrease the duration of a consultation to consumer-grade applications such as speaker identification or music genre classification for improved playlist recommendations.

The rest of the paper is structured as follows. Chapter 1 describes the theoretical notions used to implement the proposed system. Chapter 2 provides the problem description and the proposed solution as well as its implementation details. Chapter 3 offers an in depth explanation of the performed experiments and presents the corresponding results. Finally, Chapter 4 is dedicated for conclusions and future development.

Chapter 1
Theoretical Background

1.1 Signal Processing

A sound wave represents a disturbance that propagates in space through an elastic environment (such as air or water), causing the affected particles to oscillate around their equilibrium positions.

Any real world phenomenon represented by a quantity that changes with time can be viewed as a signal. For instance, the oscillation of an air particle around its equilibrium position can be viewed as a signal. An analog signal is a mathematical function with continuous values that varies over time [15].

To gain (useful) knowledge from the signals, a series of operations must be performed on the aforementioned functions. This set of possible operations is called *signal processing*.

Signal processing refers to a succession of mathematical operations applied to the signals for diverse purposes, depending on the desired field of application. These purposes may include: amplification, filtering, denoising, sampling, etc.

With the use of signal processing, signals can also be represented in various domains to emphasize their relevant features for a given application. For instance, a noisy signal viewed in the time domain might not yield substantial knowledge regarding its nature and hence might be mistaken for noise, as depicted in Figure 1.1:

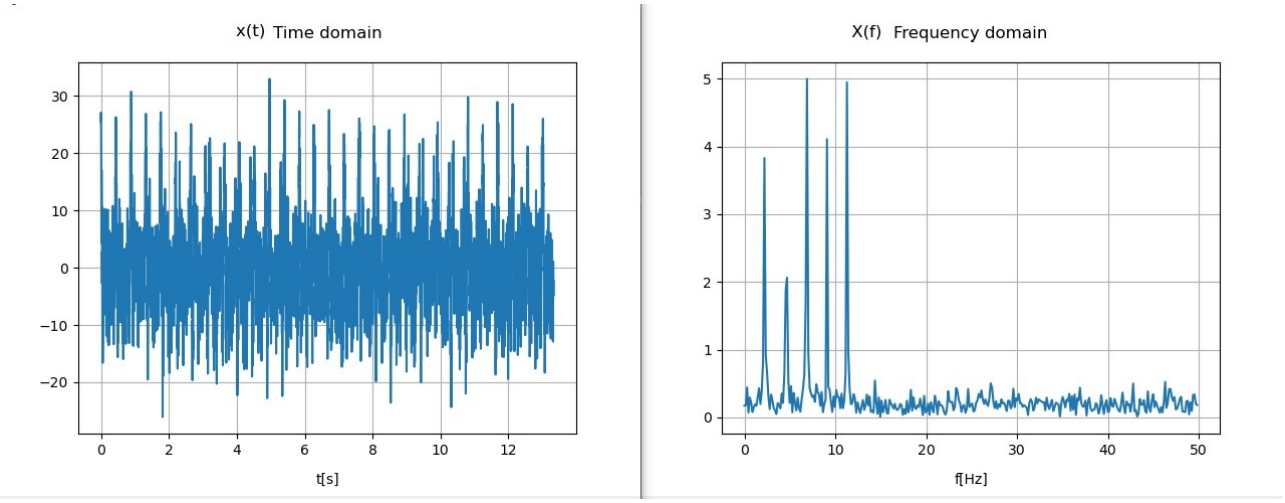


Figure 1.1: Time representation and Spectrum of a noisy signal

The frequency domain refers to the analysis of signals with respect to frequency rather than time, seen as a decomposition of the time domain signals into the amplitudes of their different sine wave frequency components and their corresponding phase shift. The phase spectrum is required along with the amplitude spectrum for the correct recovery of the original signal [16].

In the above example, the time domain signal on the left can be easily mistaken for noise, while its frequency domain representation reveals five frequency components. For such an operation, the mathematical tool employed is a transform. A Transform is a mathematical operation that decomposes a signal into its constituting base signals. In theory, a signal can be approximated with an infinite sum of any base signals depending on the properties sought after. In the case of the Fourier Transform, the base signals are represented by sinusoidal signals of varying frequencies and amplitudes due to their property of shape invariance when they pass through filters.

1.1.1 The Fourier Transform

The Fourier Transform is a linear transform that converts the amplitude over time representation of a signal,$x(t)$, into a complex function of argument $\omega = 2\pi f$ (i.e. at any given frequency, f, the signal is represented by a point in the complex plane). The decomposition of the time domain signal into its base signal components can be achieved with the use of the Fourier Transform in equation 1.1 and some additional operations that will be presented below, while the recovery of the original time domain signal from the frequency domain can be achieved with the use of the Inverse Fourier Transform as equation 1.2 shows [17].

$$X(\omega) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}d\omega \tag{1.1}$$

$$x(t) = \mathcal{F}^{-1}\{X(\omega)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega \tag{1.2}$$

However, the complex representation of the Fourier Transform does not carry much relevant physical meaning, hence it can be more conveniently represented as an amplitude and phase spectrum with the following operations:

- The amplitude spectrum, $A(\omega) = |F(\omega)|$
- The phase spectrum, $\Phi(\omega) = arctan\left(\frac{\Im\mathfrak{m}\{F(\omega)\}}{\Re\mathfrak{e}\{F(\omega)\}}\right)$,

where $\Re\mathfrak{e}\{F(\omega)\}$ is the real part of the Fourier Transform and $\Im\mathfrak{m}\{F(\omega)\}$ is the imaginary part and $F(\omega) = A(\omega) \cdot \Phi(t\omega)$.

Figure1.3 depicts the difference between the complex (i.e. Real and Imaginary part) and physical (i.e. amplitude and phase) representations of the Fourier Transform of a given signal, where the expression of the original signal is

$$x(t) = 0.25sin(2\pi 2.5t) + 0.25sin(2\pi 3.5t) + 0.25sin(2\pi 4.5t) + 0.25sin(2\pi 5.5t)$$

Most commonly in the time and frequency analysis, the following properties of the Fourier Transform exhibit the most influence in the signal's analysis: linearity, time/frequency scaling and convolution [18].

1.1.2 Linearity

The property of linearity refers to the fact that adding a two signals together or multiplying a signal by a constant in the time domain yields the same effect on their frequency domain counterparts and

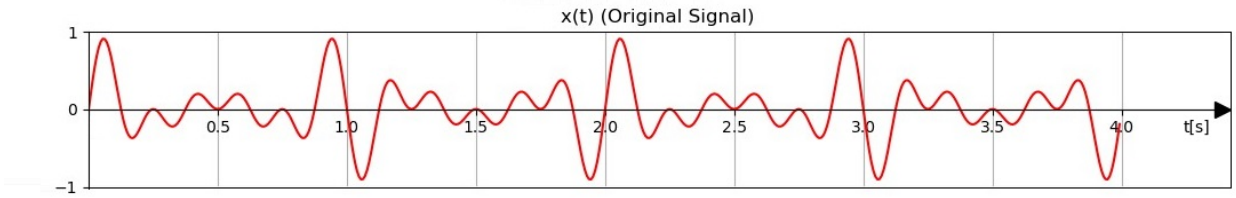


Figure 1.2: Original Signal Example

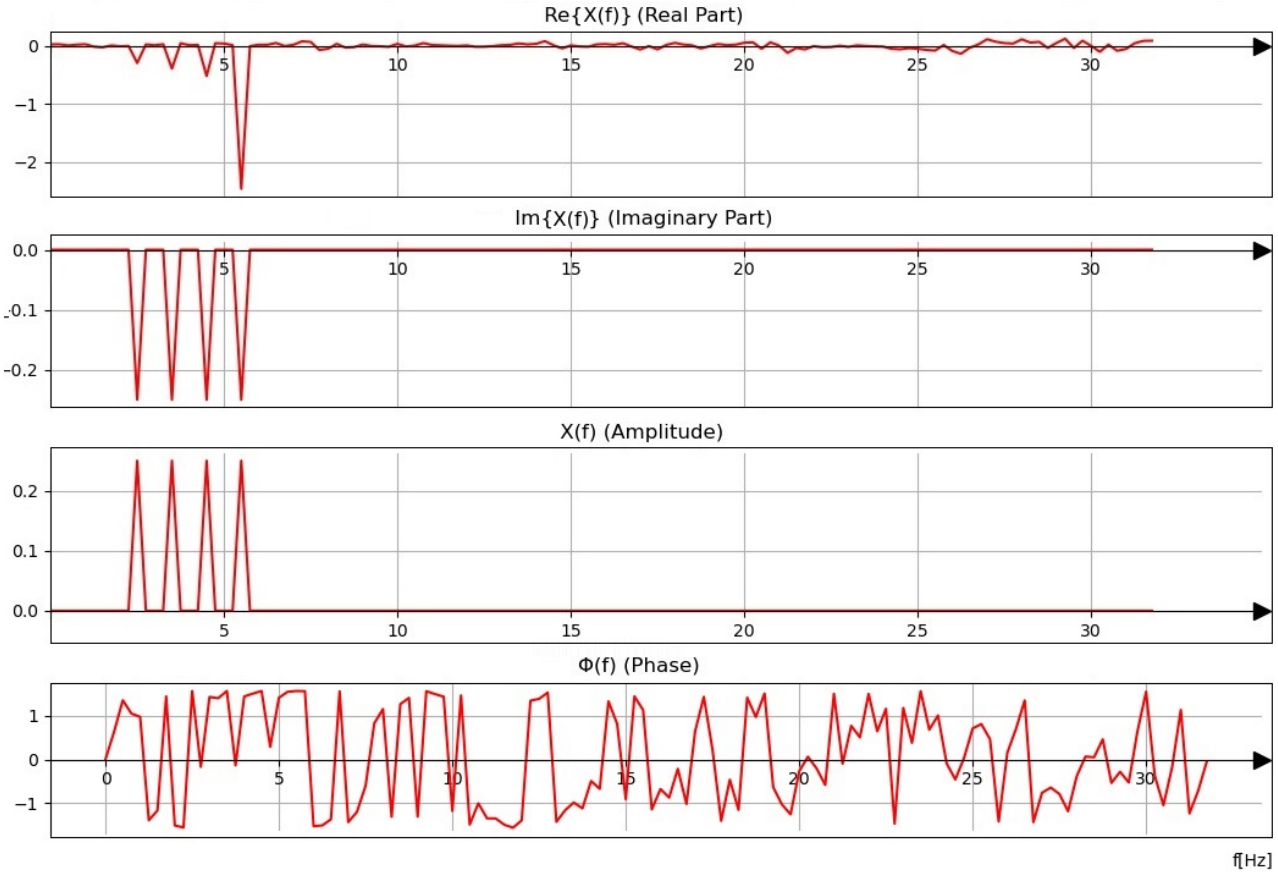


Figure 1.3: Real and Imaginary vs Amplitude and Phase representation

vice-versa.

$$ax(t) \leftrightarrow aX(\omega) \tag{1.3}$$

$$x(t) + q(t) \leftrightarrow X(\omega) + Q(\omega), \tag{1.4}$$

where a is a constant, $x(t)$ and $q(t)$ are two signals in the time domain and $X(\omega)$, $Q(\omega)$ are their Fourier Transforms.

1.1.3 Time/Frequency Scaling

The property of time/frequency scaling refers to the fact that rescaling the variable in one domain has an effect on the scale of the variable from the other domain. If the variable of one domain is scaled by a factor of q, the variable of the other domain is de-scaled by the same factor of q and the amplitude is rescaled accordingly:

$$x(qt) \leftrightarrow \frac{1}{|q|} S\left(\frac{\omega}{q}\right) \tag{1.5}$$

$$X(q\omega) \leftrightarrow \frac{1}{|q|} s\left(\frac{t}{q}\right) \tag{1.6}$$

1.1.4 Convolution

The property of convolution means that if a signal's representation is multiplied with another signal's representation in one domain (i.e. time or frequency), the resulting signal's representation will be the convolution of the two signals' representations in the other domain:

$$x(t) \cdot w(t) \leftrightarrow X(\omega) * W(\omega) \tag{1.7}$$

$$X(\omega) * W(\omega) \leftrightarrow x(t) \cdot w(t) \tag{1.8}$$

The identity element of the convolution operation is the Dirac delta function or unit impulse. Its value is zero on the real axis except at zero and its integral over the real domain equals to 1.

$$X(\omega) * \delta(\omega) = X(\omega) \tag{1.9}$$

$$\int_{-\infty}^{\infty} \delta(\omega) d\omega = 1 \tag{1.10}$$

The Fourier Transform of the identity element for the convolution operation is the identity element for the multiplication operation and vice versa:

$$\mathcal{F}\{\delta(t)\} = 1(\omega) \tag{1.11}$$

$$\mathcal{F}^{-1}\{\delta(\omega)\} = 1(t) \tag{1.12}$$

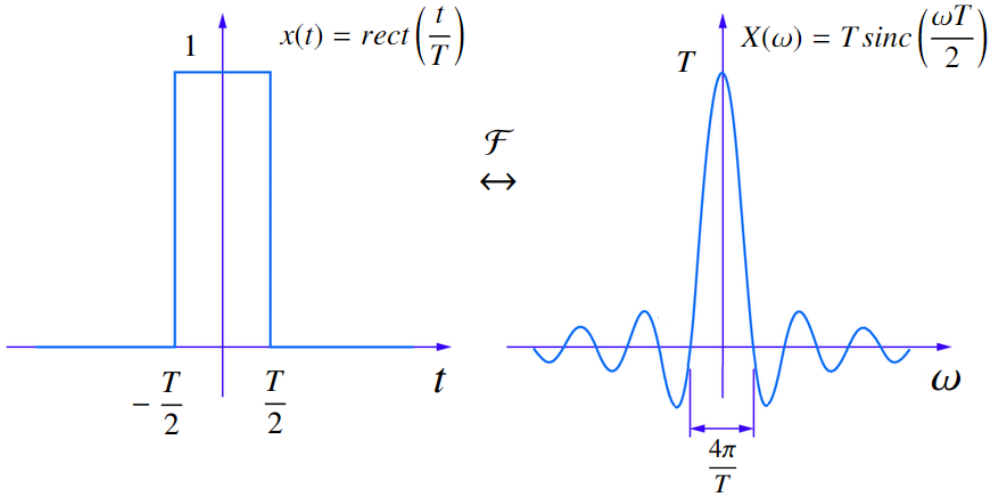
1.1.5 Windowing in the time domain

These properties can be observed as unwanted phenomena when the Fourier Transform is applied to a time window of a signal because windowing a signal [19] can be viewed as the product between the signal and the windowing function shifted with a chosen offset, Δt:

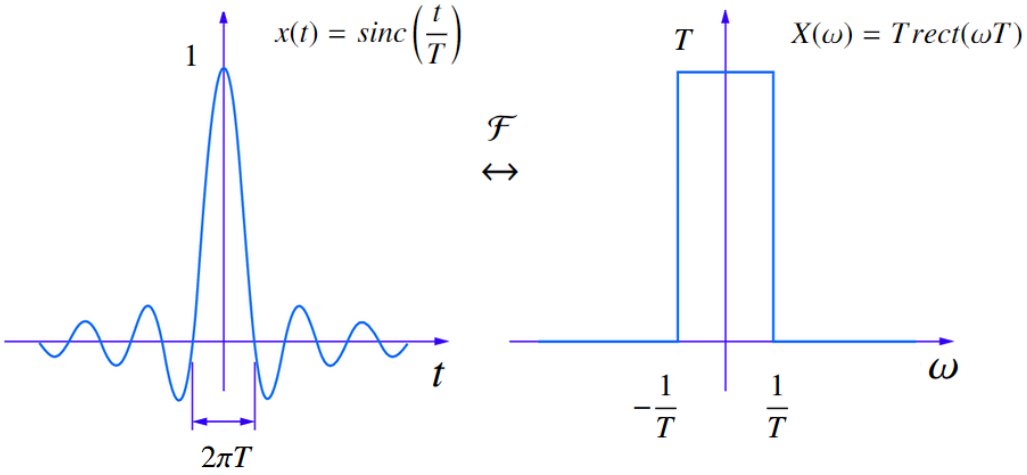
$$x(t)w(t - \Delta t) \tag{1.13}$$

Figures 1.4a and 1.4b depict the duality of the frequency and time representation of the rectangular window as well as the scaling property of the Fourier Transform. In theory, the Fourier Transform of a signal with infinite duration corresponds to the unaltered spectrum of the original signal. In practice, infinite signals are impossible, thus any finite signal can be viewed as its version of infinite duration windowed by a rectangular window of arbitrary duration.

As the width of the window increases, the width of the main lobe of its counterpart from the other domain decreases and hence it will more closely resemble a dirac delta function so its effect on the frequency spectrum of the windowed signal will be more similar to the effect of the identity element for the convolution operation.



(a) Spectrum of the rectangular window



(b) Spectrum of a cardinal sine function

Figure 1.4: Direct and Inverse Fourier Transform of a rectangular window

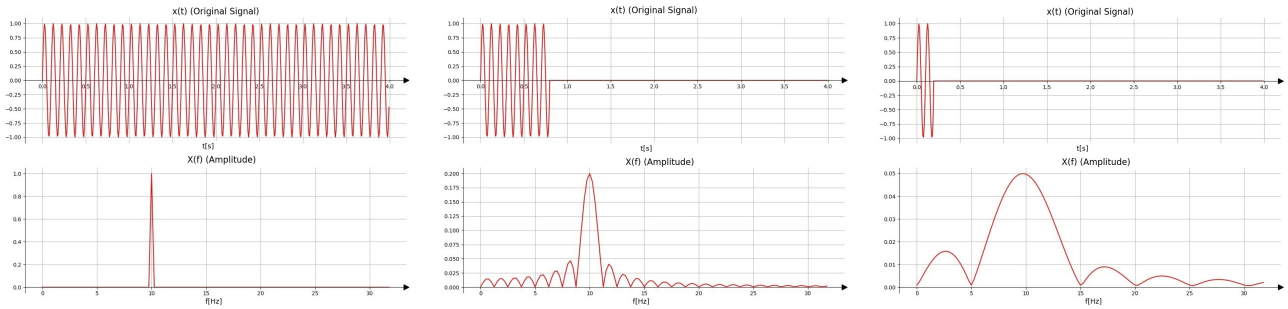
To demonstrate this effect, a sine wave was used as the original signal due to its periodicity. The signal was windowed with rectangular windows of lengths of kT, where $k \in \mathbb{N}$ and T is the period of the signal. Figures 1.5a, 1.5b and 1.5c depict the same signal with no windowing, 4 periods windowing and 1 period windowing, respectively.

Spectral leakage

Figure 1.5 also demonstrates the spectral leakage effect. In the broadest sense, spectral leakage is defined as any new frequency components or modifications to the existing frequency components, generated by any operation applied to the original signal [20].

Therefore, when short signal windows are sought for a given domain of application along with minimum alterations of the frequency spectrum, the time window's shape can be chosen to meet the required parameters. These parameters refer to the width of the main lobe of the frequency representation of the windowing signal and the amplitude of its ripples. More precisely, the width of the main lobe and the amplitude of the ripples are desired to be minimal.

Figure 1.6 depicts a selection of possible windowing functions and their corresponding amplitude spectra. These types of windowing functions are the most frequently used in practice among other



(a) 20 periods (b) 4 periods (c) 1 period

Figure 1.5: Cosine Signal Windowing Demonstration

less common windowing functions that can be used depending on the desired characteristics.

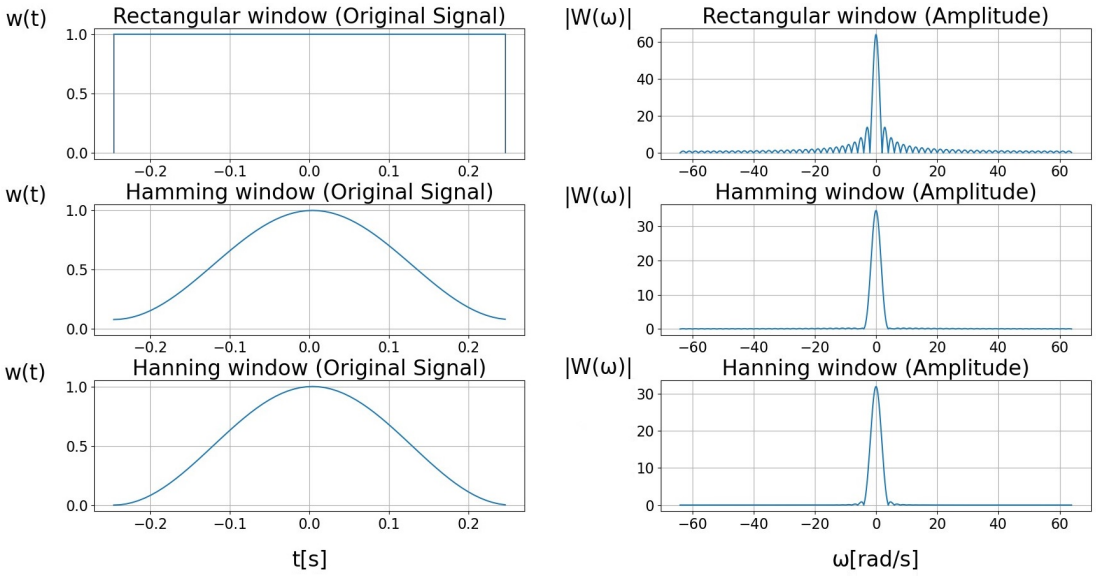


Figure 1.6: Types of windowing signals and their amplitude spectra

The mathematical expressions of these signals are the following:
Rectangular window centered around 0:

$$rect_T(t) = \frac{|(t + T/2)| + (t + T/2)}{2\,(t + T/2)} - \frac{|(t - T/2)| + (t - T/2)}{2\,(t - T/2)} \tag{1.14}$$

Where T is the parameter that specifies the width of the rectangular window.
Hamming window:

$$hamming_T(t) = rect_T(t) \cdot \left(0.54 - (1 - 0.54) \cdot cos\left(\frac{2\pi\left(t + \frac{T}{2}\right)}{T}\right)\right) \tag{1.15}$$

Hanning window centered around 0:

$$Hanning_T(t) = rect_T(t) \cdot 0.5\left(1 - cos\left(\frac{2\pi\left(t + \frac{T}{2}\right)}{T}\right)\right) \tag{1.16}$$

Hence, there will always be a trade-off between the fidelities of the amplitude spectrum of the original signal window and its time domain representation. Additionally, there will also be a trade-off

between the amplitude of the ripples and the width of the main lobe of the windowing function's amplitude spectrum. The latter phenomenon is due to the fact that the width of the main lobe varies contrarily to the ripples' amplitudes.

In addition to the the time domain analysis (i.e. manipulating the signal's variation with respect to time) and the frequency domain analysis (i.e. frequency spectrum obtained with the Fourier Transform), signals can be analyzed in other domains (i.e. Time-Frequency domain, Mel-Frequency Cepstral domain, Cepstral domain, Wavelet and many more).

1.1.6 The Time-Frequency Representation

To represent the variation of the amplitude spectrum over time for a given signal , the Short Time Fourier Transform or STFT can be used. The STFT represents a series of Fourier Transforms applied to subsequent, disjoint or overlapping time windows of the original signal. Each time window has the same duration, τ also known as the analysis frame. The time windows are extracted with the use of a windowing function of width τ, shifted with a number of analysis frames and are represented with the Fourier Transform. The plot of an STFT's amplitude spectrum is known as a spectrogram.

$$STFT_{\tau,hop}\{x(t)\}(T,\omega) = X(T,\omega) = \mathcal{F}\{x(t)w(t-T-\tau/2)\}; T \in \{k \cdot hop, k \in \mathbb{N}, hop \in (0,\tau]\} \tag{1.17}$$

Where τ is the duration of a window and T is the time variable of $X(T,\omega)$. The choice of the *hop* parameter can be used to adjust the percentage of how much two subsequent signal window overlap. If the parameter is set to τ, the windows are disjoint. If the parameter is set close to 0, the windows are almost perfectly overlapped.

1.1.7 STFT time and frequency resolution trade-offs

Frequency resolution refers to the separability of two frequency components that are close together, while time resolution refers to the moment when a signal changes its frequency in time. Due to the scaling property of the Fourier Transform, the trade-off between time and frequency resolution arises: As the frequency resolution increases, the time resolution decreases and vice versa. This effect is demonstrated in figures 1.7 and 1.10 where two original signals (with note durations of 125 and 62.5 milliseconds) were windowed by non-overlapping hanning windows of approximately 6, 24 and 96 millisecond widths.

The original signal represents stream of piano notes A3 and F4 (of fundamental frequencies 220 Hz and 350 Hz approximately) that last 125 milliseconds each. In figure 1.7, the individual notes' amplitude spectra can be observed as a heat map:

When comparing figures 1.8 a and b that depict the initial notes' amplitude spectra, the fundamental frequencies of the two notes and their higher harmonics can be clearly distinguished, as the two figures were obtained from the individual notes lasting 14 seconds:

Figure 1.7 depicts the spectrogram of the above mentioned original signal. The signal was windowed with a hanning window and the Mel scale was used for the frequency axis and the analysis window length of approximately 24 milliseconds. The spectrograms were created with the Audacity audio editing software [21].

The Mel Scale

The Mel (name inspired by the word "melody") scale is a scale of pitches, based on the human auditory system that exhibits a logarithmic sensitivity to sinusoidal components of varying pitches

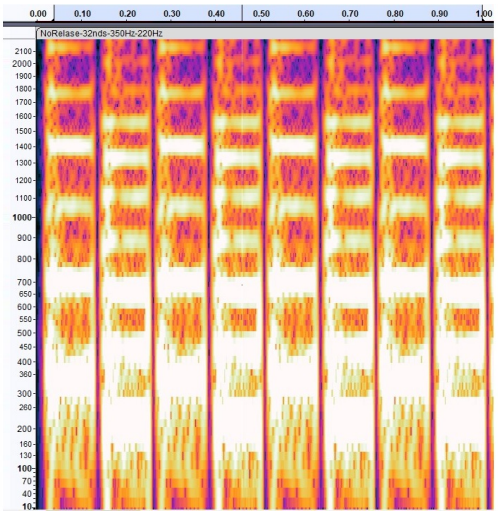
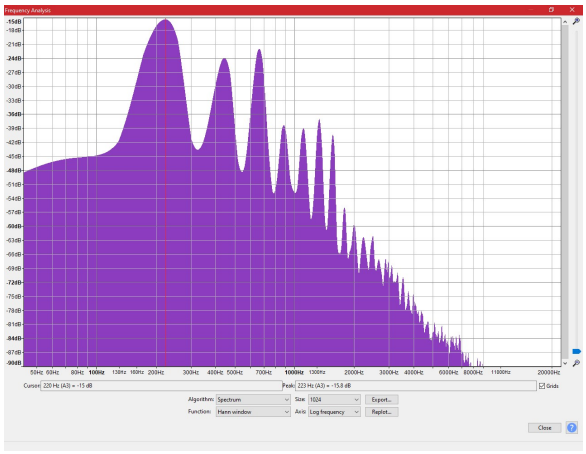
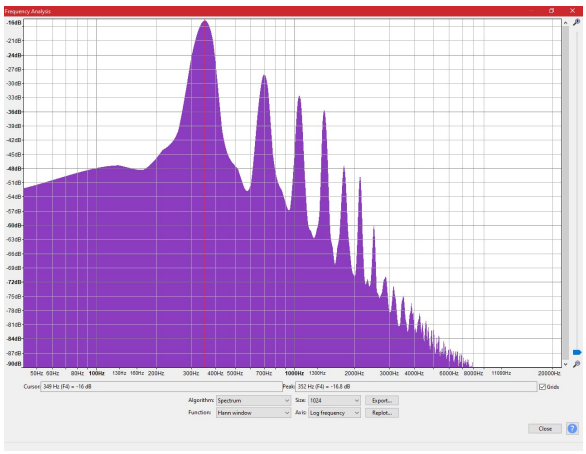


Figure 1.7: Spectrogram of 125 ms lasting notes



(a) A3 frequency spectrum



(b) F4 frequency spectrum

Figure 1.8: Frequency spectra of initial notes

and constant amplitude. There is no standard conversion formula between linear frequency and Mel-scaled frequency, most likely because each person's auditory system exhibits a different sensitivity characteristic, however O'Shaughnessy's conversion formula is one of the most commonly used:

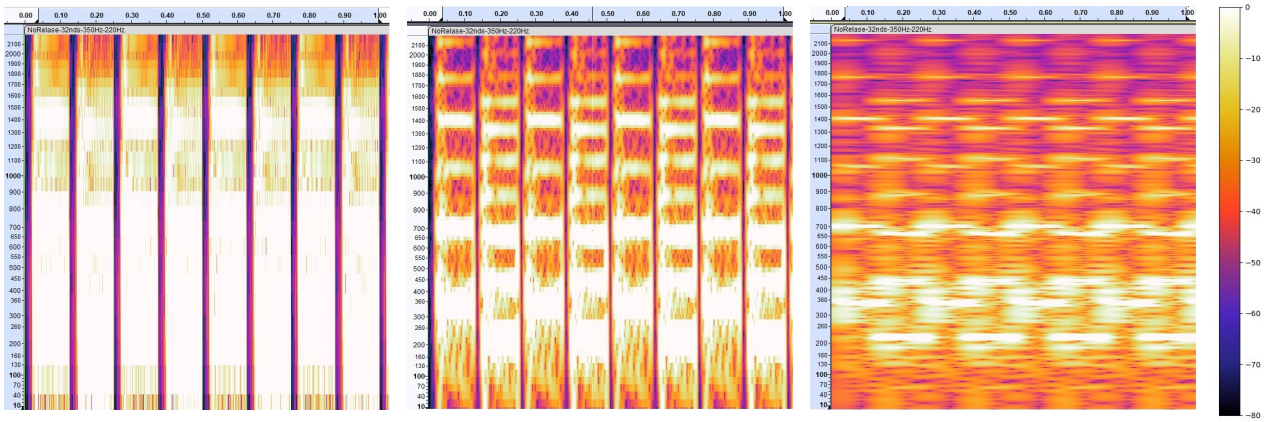
$$m = 1127ln\left(1 + \frac{f}{700}\right) \tag{1.18}$$

The inverse expression to retrieve frequency from the Mel representation is:

$$f = 700\left(e^{\frac{m}{700}} - 1\right) \tag{1.19}$$

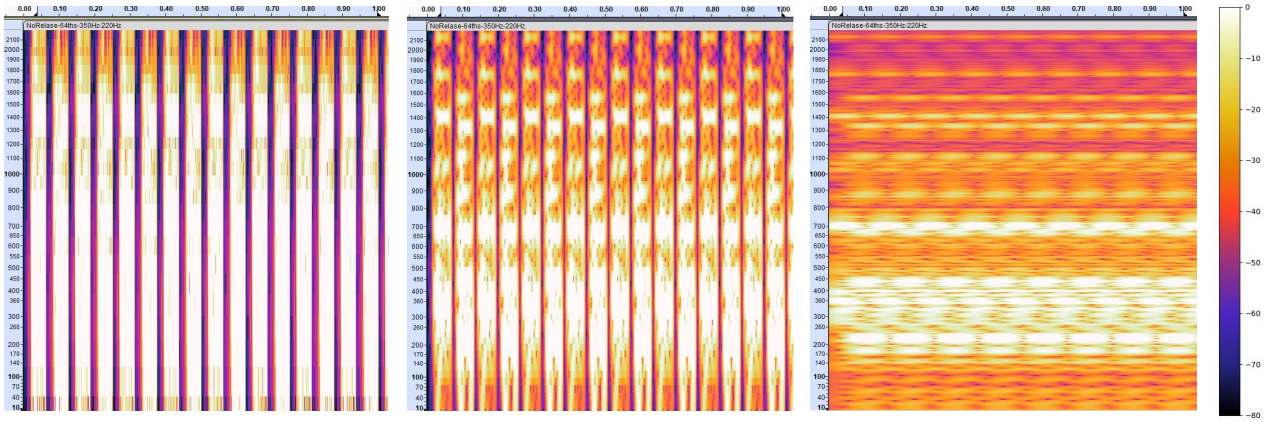
In figure 1.9 c) two subsequent notes are represented in one time region and the spectrum of a note is distorted by the presence of the other note in the said time region, while in figure 1.9 a, each note's fundamental frequency is inseparable from its higher frequency components.

Both figures 1.9 and 1.10 have been obtained with non-overlapping hanning windows and the Mel scale. The heatmap's values are depicted in decibels, on the rightmost column of both figures. The comparison between the two figuers demonstrates that the resolution trade-off becomes more sensible as the original signal exhibits a faster variation. Figures 1.9b and 1.10b show that the frequency components of each second note (i.e. A3) from the signal with faster variation are indistinguishable and



(a) 6 ms analysis window (b) 24 ms analysis window (c) 96 ms analysis window

Figure 1.9: Resolution trade-off for 125 ms lasting notes



(a) 6 ms analysis window (b) 24 ms analysis window (c) 96 ms analysis window

Figure 1.10: Resolution trade-off for 62.5 ms lasting notes

appear to have merged into a wider frequency band centered at 300 Hz. This representation is inaccurate as figure 1.8a indicates that A3's fundamental and first harmonic are centered at approximately 220 Hz and 450 Hz.

If figures 1.9a and 1.10a shown a clear distinction between each note onset via the presence of the darker columns between each note, figures 1.9c and 1.10c show that the onsets of two subsequent notes are no longer clearly distinguishable because of the length of the analysis window, as well as the fact that their spectra overlap.

Hence, the time-frequency resolution trade-off is not only influenced by the choice of the STFT analysis window, but also by the signal's variation speed.

For an easier view, the spectrograms can be represented using different (i.e. more relevant to the human hearing system) frequency scales, namely: the logarithmic scale or the Mel scale. In the spectrogram figures, the Mel scale was used.

A spectrogram with the Mel scale on the frequency axis is called a Mel spectrogram.

1.1.8 The Cepstral Domain Representation

The cepstral domain representation is obtaind by applying the Cepstrum transform to the original signal. The name Cepstrum results from the word "spectrum" with the first four letters reversed. The variable of the cepstral domain is represented by quefrency which was similarly obtained by rearranging the letters of "frequency". In equation 1.20, b represents the base of the logarithm applied

to the Fourier Transform of the original signal, x(t), and its choice is user-dependant. Equations 1.20, 1.21 and 1.22 define the complex, real and power cepstrum respectively.

$$C_c\{x(t)\} = \mathcal{F}^{-1}\{log_b(\mathcal{F}\{x(t)\})\} \tag{1.20}$$

$$C_r\{x(t)\} = \mathcal{F}^{-1}\{log_b(|\mathcal{F}\{x(t)\}|)\} \tag{1.21}$$

$$C_p\{x(t)\} = \left|\mathcal{F}^{-1}\left\{log_b\left(|\mathcal{F}\{x(t)\}|^2\right)\right\}\right|^2 \tag{1.22}$$

The Mel-frequency Cepstral Representation will be discussed in the following section.

1.2 Digital Signal Processing

Digital signal processing is the mathematical manipulation of digital signals and therefore requires the signals to be digitally interpreted as a time series (i.e. digital signal). The necessity of digital interpretation is due to the complexity operations needed to extract information from the signals and the necessity to store the time series that are otherwise impossible. Figure 1.11 illustrates the sampling process of an analog signal.

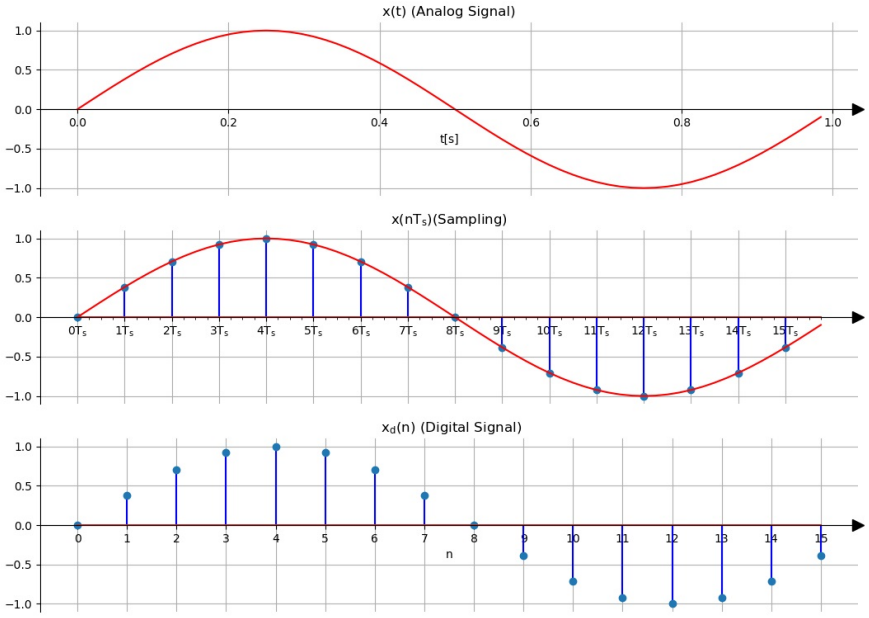


Figure 1.11: Sampling Process

Digital signals (i.e. discrete signals) can be obtained by sampling continuous signals. Sampling is the process of measuring the amplitude of the analog signal (that can also be viewed as a signal with an infinitely dense number of samples over an arbitrarily short time interval) at different moments in time. The measured values are then quantized and stored digitally as approximate values of the measured quantity at a given moment in time. The precision of such an approximation is determined by the number of binary memory cells (i.e. bits) that were used to represent the value of the quantity [22]. Ideally, a real number requires an infinity of memory cells to be represented with no error. Figure 1.12 depicts the block diagram of an Analog to Digital Converter (ADC). Equations 1.23 and 1.24 depict the mathematical formalisation of the sampling process, where n is the discrete variable

denoting the index of the current sample and $\delta(t-nT_s)$ is the dirac delta function shifted by n sampling periods, nT_s [23].

$$x(nT_s) = \sum_{n=-\infty}^{\infty} x(t)\delta(t - nT_s) \tag{1.23}$$

$$x_d(n) = x(nT_s) \tag{1.24}$$

The digital signal $x_d(n) = x(nT_s)$, where $T_s = \frac{1}{F_s}$ and $x(t)$ is the analog signal.
For a correct digital representation of the analog signal, the Nyquist-Shannon theorem [24] states that the analog signal's amplitude values must be sampled at a frequency (sampling rate, F_s) of at least twice the value of the highest frequency of interest, $2F_M$ (i.e. the Nyquist frequency), from the spectrum of the input signal, to avoid the aliasing of the useful spectrum. Aliasing is a type of distortion that causes the original signal to become unrecoverable after being sampled [25]. In the frequency domain, this phenomenon translates to an overlap of the original signal's frequency spectrum periods. The overlapping regions are summed together and result in the aforementioned distortion.

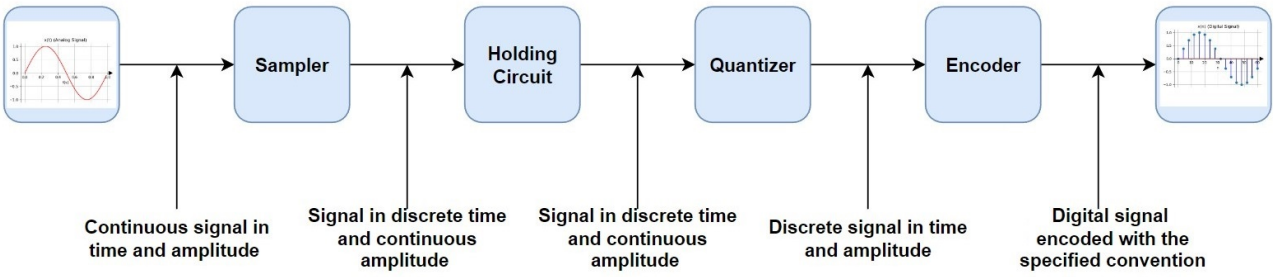


Figure 1.12: Block Diagram of an Analog to Digital Converter

1.2.1 The Discrete Time Fourier Transform

Discrete Signal Processing Notations

- F – the frequency of the analog signal, $x(t)$
- Ω – the angular frequency variable, $2\pi F$, in the analog domain
- Ω_0 – the angular frequency, $2\pi F_0$, of the analog signal, x(t)
- F_s – the analog sampling frequency
- $\Omega_s = 2\pi F_s$ – the analog sampling angular frequency
- $f = \frac{F}{F_s}$ – normalized frequency
- $\omega = 2\pi f$ – normalized angular frequency variable
- $\omega_0 = 2\pi f_0$ – normalized angular frequency of the digital signal
- $\mathcal{F}\{x(nT_s)\} = X_d(\Omega)$

- $\mathcal{F}\{x(t)\} = X(\Omega)$
- N – the number of samples where the signal $x(n)$ has been sampled, with n ranging from 0 to $N-1$

After sampling, the Fourier Transform becomes the Discrete Time Fourier Transform (DTFT) and the Inverse Discrete Time Fourier Transform (IDTFT) as shown in 1.25 and 1.26:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{-jn\omega} = TFTD\{x(n)\}(\omega) \tag{1.25}$$

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) \cdot e^{jn\omega} d\omega = IDTFT\left\{X(e^{j\omega})\right\}(n) \tag{1.26}$$

The demonstration of spectrum periodization due to sampling is shown in equations 1.27 and 1.28 as follows:

$$\mathcal{F}\{x(nT_s)\} = \mathcal{F}\{x(t)\} * \mathcal{F}\left\{\sum_{n=-\infty}^{\infty} \delta(t - nT_s)\right\} \tag{1.27}$$

$$X_d(\Omega) = \frac{1}{2\pi} X(\Omega) * \Omega_s \sum_{k=-\infty}^{\infty} \delta(\Omega - k\Omega_s) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X(\Omega - k\Omega_s) \tag{1.28}$$

Likewise, the formula for the spectrum of the signal will have the expression in 1.29:

$$X_d(F) = \frac{1}{2\pi} \sum_{-\infty}^{\infty} X(F - nF_s) \tag{1.29}$$

Hence, the graphical representation of the spectrum in figure 1.13 will look like figure 1.14 if the signal is correctly sampled:

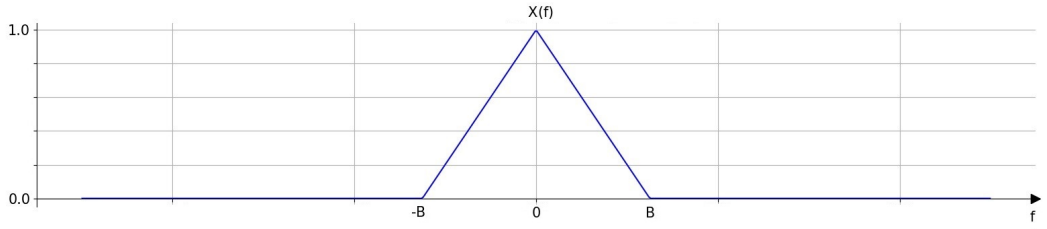


Figure 1.13: Example spectrum of an analog signal

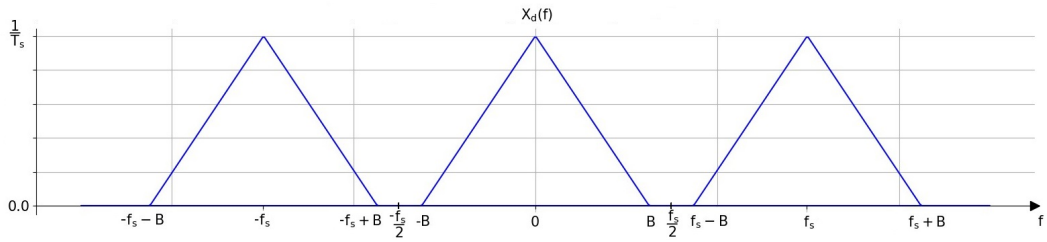


Figure 1.14: Spectrum of the digital signal

Figure 1.15 depicts the spectrum of a signal sampled with the Nyquist frequency and figure 1.16 shows the spectrum of an aliased signal sampled with a frequency below the Nyquist limit. The spectrum of a digital signal is periodic and theoretically infinite [22].

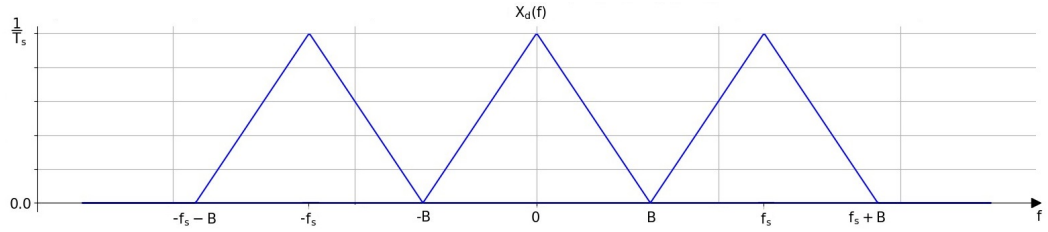


Figure 1.15: Spectrum of the digital signal sampled at Nyquist frequency

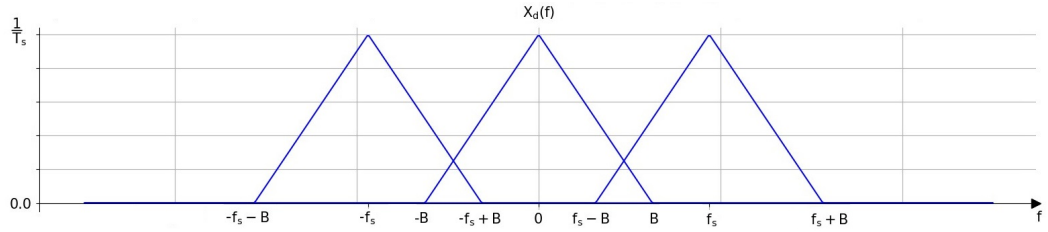


Figure 1.16: Spectrum of the signal sampled below Nyquist frequency

As the sampling rate increases, the digital signal better approximates the analog signal, where sampling with $F_s = \infty$ returns the analog signal, since the analog signal is an infinite sequence of points [22].

For a computationally feasible implementation, the DFT will be applied using N as the number of points to compute the Fourier transform as shown in equation 1.30:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}k} = DFT\left\{x(n)\right\}(k) \tag{1.30}$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) \cdot e^{j\frac{2\pi}{N}k} = IDFT\left\{X(k)\right\}(n), \tag{1.31}$$

where k is the variable denoting the indices of each frequency component of the DFT.

In practice, equation 1.32 is used to compute the DFT in a user-specified number of N_{DFT} points:

$$X(k) = \sum_{n=0}^{N_{DFT}-1} x\left(n\frac{N}{N_{DFT}}\right) \cdot e^{-j\frac{2\pi}{N_{DFT}}k} = DFT_{N_{DFT}}\left\{x(n)\right\}(k) \tag{1.32}$$

Conceptually, the DFT corresponds to a sampling of N_{DFT} equidistant points from the DTFT's spectrum, because the calculation of integrals is impossible since computers can only operate with discrete values. Hence, equations 1.30 and 1.31 derive from the DTFT's normalized angular frequency variable, ω, that becomes $\omega \rightarrow \omega_k = \frac{2\pi}{N}k$, integration is replaced by summation and $N-1$ will correspond to ω_s.

Real digital signals are also finite in duration, hence no truly periodic digital signals exist. Therefore, the digital representation of a sine wave is just the product of a digital rectangular window of length N and the sampled sine wave. The digital rectangular window's expression is depicted in equation 1.33 and its visual representation is illustrated in figure 1.17, where D represents the length of the rectangular window.

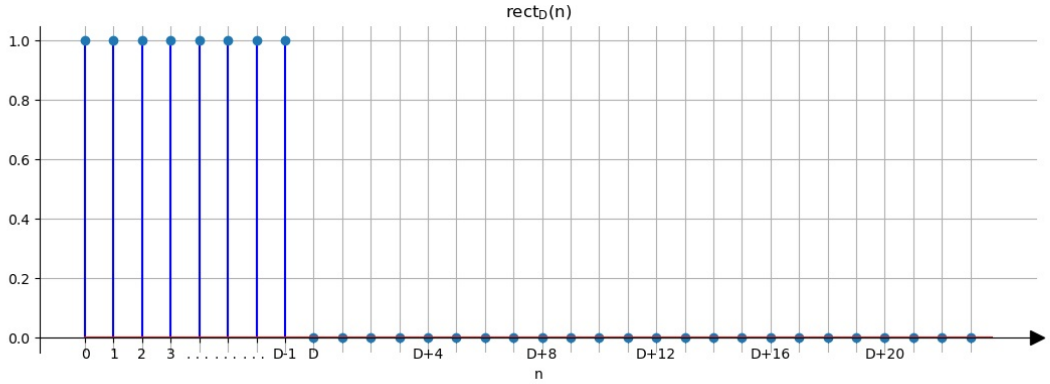


Figure 1.17: Digital rectangular window of length D

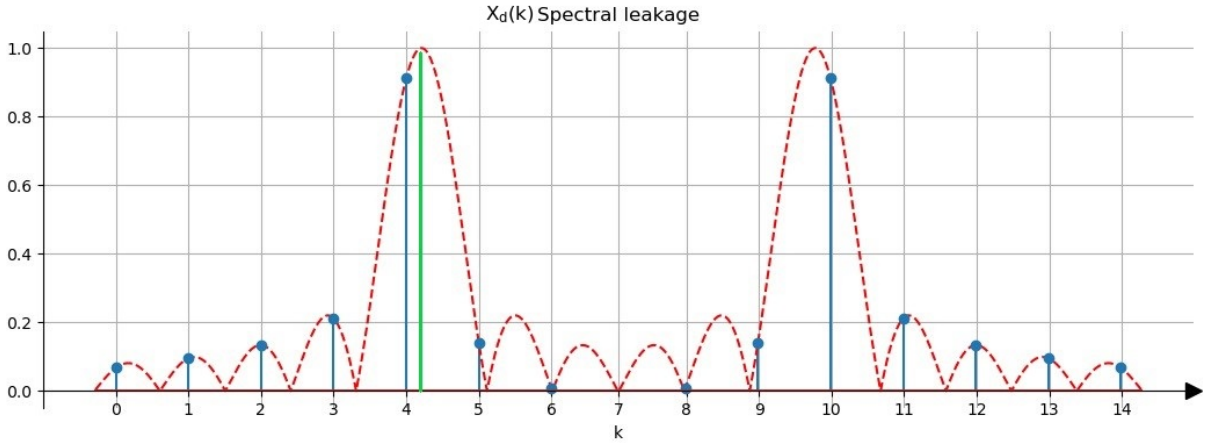


Figure 1.18: Digital spectral leakage

$$rect_D(n) = \frac{|n| + n}{2n} - \frac{|n - D| + (n - D)}{2(n - D)} \tag{1.33}$$

To demonstrate the phenomenon of spectral leakage caused by the DFT, the spectrum of a windowed discrete sine wave, $x(n) = Asin(\omega_0 n)rect_D(n)$ will be used. If the angular frequency of the sine wave, ω_0 is not an integer multiple of $\frac{2\pi}{N_{DFT}}$, the DFT of the signal, will introduce additional spectral leakage as shown in figure 1.18, where $F_s = 128Hz$ and the component of normalized frequency, ω_0 of the signal $x(t) = sin(40t)$ is highlighted in green.

In figure 1.18, $N_{DFT} = 15$ was chosen. Given $\omega_0 = \frac{2\pi F_0}{F_s} = \frac{2\pi}{N_{DFT}}k$, the equation becomes $\frac{40}{128} = \frac{k}{15}$, hence $k = 4.68 \notin \mathbb{Z}$.

The spectral leakage, in this case, is given by the fact that the components of the signal's DFT amplitude spectrum are slightly shifted from their positions in the amplitude spectrum of the DTFT of the same signal. Moreover, the maximum amplitude components of the DFT differ from the maximum amplitude of the DTFT.

1.2.2 Digital Representation of the STFT

In the digital world, the expression of the spectrogram becomes

$$S = |STFT_{l,hop,N_{DFT}}\{x(n)\}(bin, k)|, \tag{1.34}$$

where l is the length of the windowing function used, hop is the time step that determines the

overlap between two subsequent time windows, *bin* is the index of the time interval on which the STFT was computed and $STFT_{l,hop,N_{DFT}}\{x(n)\}(bin,k)$ is calculated using the DFT in N_{DFT} points.

1.2.3 Digital Representation of the Cepstrum Transform

In the digital world, the Cepstrum Transform remains similar to its continuous counterpart, with a single difference: The digital Cepstrum allows an additional user-specified parameter of N_{DFT} points to compute the DFT. Equations 1.35, 1.36 and 1.37 define the digital complex, real and power cepstrum, respectively.

$$C_{cN_{DFT}}\{x(n)\} = IDFT\{log_b(DFT_{N_{DFT}}\{x(n)\})\} \tag{1.35}$$

$$C_{rN_{DFT}}\{x(n)\} = IDFT\left\{log_b\left(\left|DFT_{N_{DFT}}\{x(n)\}\right|\right)\right\} \tag{1.36}$$

$$C_{pN_{DFT}}\{x(n)\} = \left|IDFT\left\{log_b\left(\left|DFT_{N_{DFT}}\{x(n)\}\right|^2\right)\right\}\right|^2, \tag{1.37}$$

where b is the base of the logarithm used in the computation of the Cepstrum and can be specified by the user, depending on the application domain.

1.2.4 The Mel-frequency Cepstral Representation

The Mel-Frequency Cepstrum is a short-time representation that involves the discrete cosine transform of the logaritm of the Mel scaled power spectrogram of a signal. The base of the logarithm is user-specified and the spectrogram can be obtained with either triangular or cosine (i.e. Hamming, Hanning or other related windows) overlapping windows.

The Discrete Cosine Transform

The Discrete Cosine Transform (or DCT) is the representation of a digital signal as the terms of a sum of cosine functions of different frequency arguments [26]. The values of the DCT are real numbers. One key role of the DCT in the domain of digital signal processing is its usage in data compression due to its good energy compaction property [27]. For example, images can be compressed with ratios up to 200 : 1, with a Peak Signal-to-Noise Ratio (PSNR) of about 30 dB using the DCT in the JPEG2000 algorithm [28]. The expression of PSNR is defined in equation 1.38:

$$PSNR = 10log_{10}\left(\frac{max(x(n)}{MSE(x(n),k(n))}\right), \tag{1.38}$$

where $k(n)$ is the compressed signal and $MSE = \frac{1}{N}\sum_{n=0}^{N-1}|x(n)-k(n)|^2$ and N is the number of values in both signals x and k.

There are several variants of the DCT, of which the following four are the most common:

$$x_k = \frac{1}{2}(x_0 + (-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n cos\left[\frac{\pi}{N-1}nk\right], \quad \forall\, k \in \{0,1,...,N-1\} \tag{1.39}$$

$$x_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], \quad \forall\, k \in \{0,1,...,N-1\} \tag{1.40}$$

$$x_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)n\right], \quad \forall\, k \in \{0,1,...,N-1\} \tag{1.41}$$

$$x_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right], \quad \forall\, k \in \{0,1,...,N-1\} \tag{1.42}$$

Equation 1.43 shows the computation of the MFCC matrix:

$$MFCC\{x(n)\}(bin,k) = DCT\{log_b(|STFT_{l,hop,N_{DFT}}\{x(n)\}(bin,k)|^2)\}, \tag{1.43}$$

where k indicates the index of a bin on the Mel scale.

Figure 1.19 illustrates an example representation of MFCCs extracted from a 14s signal, sampled at 44.1kHz, of alterating A3 and F4 notes of 0.5s durations each. Each bin of the MFCC matrix contains 13 coefficients and was calculated from a 1024 sample long time frame with 0% overlap.

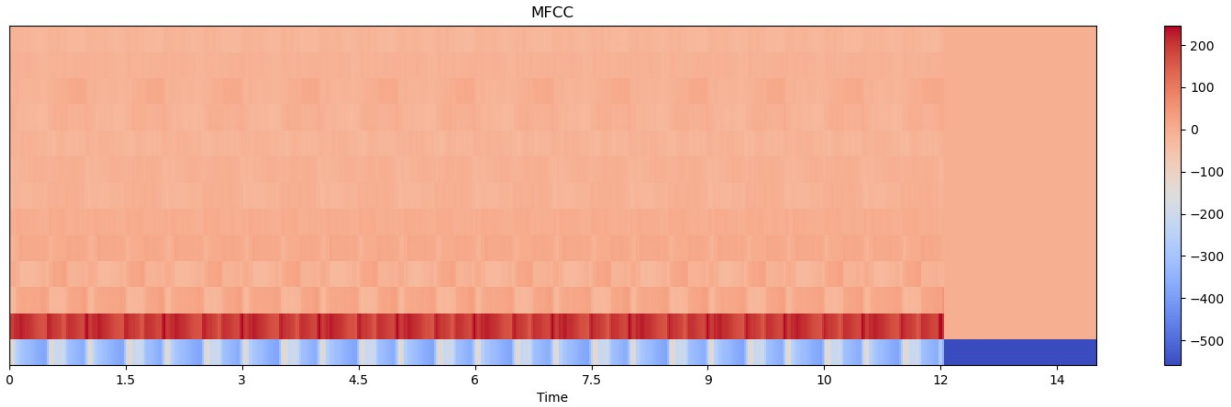


Figure 1.19: MFCCs of the notes A3 and F4

1.2.5 Other Digital Signal Features

The Zero-Crossing Rate

The Zero-Crossing Rate (ZCR) of a signal is the number of sign changes during an interval of that signal, dividied by the length of that interval, as show in the following equation:

$$ZCR = \frac{1}{2N}\sum_{n=-0}^{N-1} |sgn(x(n) - sgn(x(n-1))|\,, [29] \tag{1.44}$$

where $sgn(n) = \frac{|n|}{n}$

The Energy of a Digital Signal

The energy of a digital signal is defined by the following formula:

$$E = \sum_{n=-\infty}^{\infty} x(n)^2 \tag{1.45}$$

The Autocorrelation Function of a Digital Signal

The autocorrelation function of a signal describes the amount of correlation between two values of the same signal, situated at different time intervals. In essence, autocorrelation measures how the version of a signal, shifted with k samples, relates to itself [30]. If the shift, k is zero, the autocorrelation function indicates the energy of the signal.

The definition of the autocorrelation function is shown in the following equation:

$$R_n(k) = \sum_{k=-\infty}^{\infty} x(n)x(n-k) \tag{1.46}$$

1.3 The Human Vocal Tract

The most widely used approach of simulating the vocal tract is the source-filter model [31]. This type of representation of the vocal tract was proven effective as voice synthesis was achieved using this model [32, 33].

This approach views the vocal tract as a combination of a sound source (i.e. the vocal chords) and a composite acoustic filter that is comprised of multiple mobile and immobile parts that the act as filters for the sound. The mobile parts are also known as articulators. To name a few:

1. the larynx – acts as an air flow regulator during breathing and is useful for the protection of the airway during swallowing, contains the vocal folds
2. the glottis – acts as a valve that closes to protect the lungs when swallowing and plays a role to produce vowels and certain consonants
3. the epiglottis – possesses a similar role in speech to the glottis and is also involved in whisper [34]
4. the pharynx – an echo chamber with variable size that is controlled by the lower part of the tongue
5. the uvula – role in enunciating some consonants
6. the vleum – contains muscular fibers responsible for closing the nasal cavity; also known as the soft part of the palate
7. the oral cavity – acts as a compound resonator of variable shape and size due to the adjustability of the relative positions of the: teeth, palate, lips and tongue
8. the tongue – acts as an articulator and due to its mobility earns the primary role in speech and sound formation as it enables the pronounciation of more than 100 words per minute
9. the hard palate – aids the togue's articulation of certain sounds
10. the alveolar ridge – aids the togue's articulation of certain sounds
11. the teeth – aid the togue's articulation of certain sounds
12. the lips – articulator that changes the resonance of the oral cavity for different sounds; also plays a role in enunciating a type of consonants called plosives
13. the nasal cavity – acts as a resonator that allows the pronounciation of nasal consonant sounds that are produced when the velar muscles leave the nasal cavity open

Figure 1.20 depicts the position of the above-mentioned parts of the vocal tract.

For speech production, the accurate coordination of all the elements enumerated above is required [35]. Such coordination implies significant synchronization adjustments between multiple elements of the human phonatory system [36]. This function is controlled by multiple areas in the brain [37].

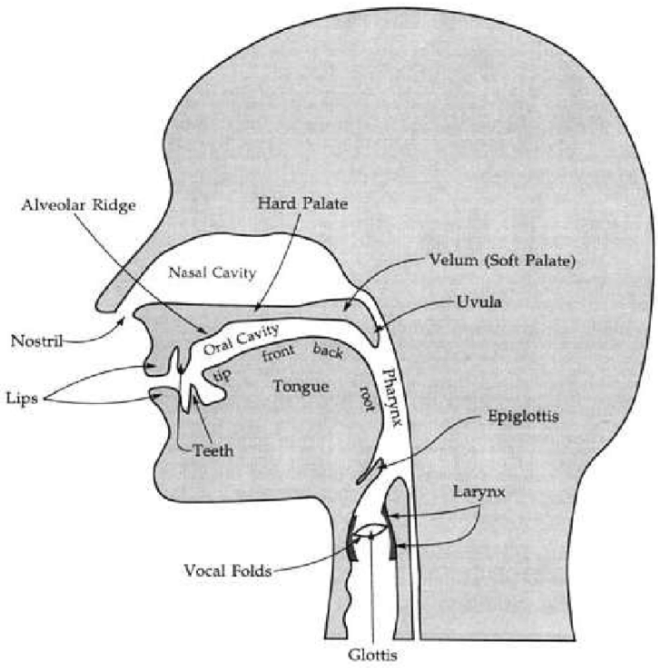


Figure 1.20: Midsagital cross-section of the Vocal Tract [1]

1.3.1 Particularities of the Vocal Signal

The speech signal or the vocal signal is represented by a rapidly changing, aperiodic sequence of sounds (known as phonemes). Therefore, the standard analysis methods based on the assumptions of signal stationarity or slow variation can not be applied [38].

Depending on the nature of each sound's production mechanism, the vocal signal exhibits different features. The vocal signal is quasistationary on windows of approximately 20 milliseconds [39]. Additionally, the fundamental pitches of each vocal signal depends on the speaker. The mean fundamental pitch values are between 85-155 Hz, 165-255 Hz and 250-400 Hz for males, females and children, respectively [40, 41]. The vocal signal consists of phonemes. A phoneme is the shortest phonetic unit that carries meaning in speech. A phoneme is the equivalent of a letter in written language [42]. Phonemes can be divided into two major categories: vowels (i.e. *a*, *e*, *i*, *o*, *u*) and consonants. In the English language, consonants can be further divided as follows [43]:

- nasal (*m*, *n*) – these require the nasal cavity to be connected with the oral cavity when produced
- unvocalized fricatives (*f*, *th*) – produced by the friction of breath passing through a narrow space (such as an opening between the tongue and the teeth or the palate)
- vocalized fricatives (*v*, *z*) – produced similarly to unvocalized fricatives but also employ the vocal folds
- unvocalized plosives (*c*, *p*, *t*) – produced by the sudden relase of pressure after being relased from an airtight constriction of the oral cavity (for instance, between the tongue and the alveolar ridge)
- vocalized plosives (*b*, *d*, *g*) – produced similarly to unvocalized plosives but also employ the vocal folds
- glides or semivowels (*y* in "toy", *w* in "how") – produced like two vowels pronounced jointly by changing the position of an articulator during the pronounciation of the first vowel that results in a swift transition between the two composing sounds

- liquids (*l*, *r*) – produced by the resonance of a vowel-like consonant that originates from a partial closure in the oral cavity
- affricates or semiplosives (*j*, the *"ch"* sound) – produced by opening an airtight stricture, like plosives, but relasing a fricative

Features of different phonemes

Each phoneme is understood and distinguished by the human auditory system based on a series of features that characterize each sound, hence the importance of feature extraction in the processing of the vocal signals. The mathematical representation of the features recognized by the human auditory system might differ from the mathematical definition of the commonly used features in signal processing. However, such features have been successfully used for a variety of applications, including (but not limited to): text to speech, speech recognition and speech synthesis [32]. Some of the basic and easier to understand features of different phonemes can be derived by calculating their autocorrelation, energy, zero crossing rate or spectrogram to name a few. The frequency peaks with high energy concentration in the spectrum of a vocal signal are known as formants.

In comparison to consonants, vowels are characterized by periodicity, a higher energy with especially proeminent formants, and a lower zero-crossing rate.

Figure 1.21 depicts the time representation of 20 millisecond windows from the *'a'* and *'s'* phonemes, while figure 1.22 represents the autocorrelation function of the said signals:

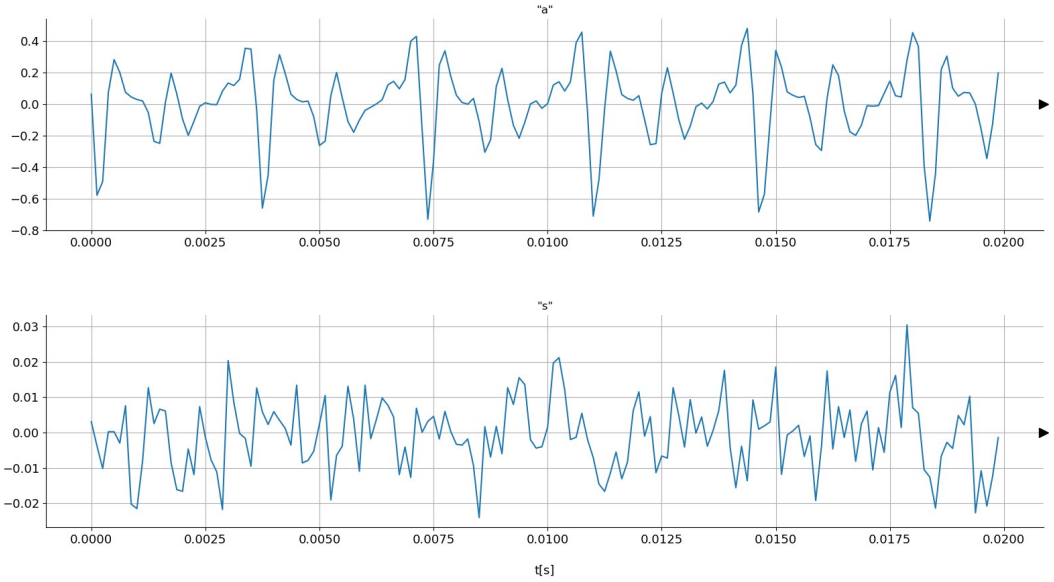


Figure 1.21: Time representation of the *'a'* and *'s'* phonemes

Figure 1.21 indicates that the time representation of *'a'* is a periodic signal and *'s'* exhibits a noise-like behaviour, while figure 1.22 clearly shows the periodicity of the signal 'a', because the autocorrelation function is periodic with the same period of the autocorrelated signal if the signal is periodic. The value of the autocorrelation function at 0 shift corresponds to the signal's energy and therefore, figure 1.22 demonstrates that the consonant's energy is almost 3 degrees of magnitude (i.e. 1000 times) lower than the energy of the vowel.

Figure 1.23 depicts the spectrograms of the two analyzed phonemes. The spectrogram of the vowel displays higher peak values and a higher variability of the amplitude values shown. This means that the formants are more proeminent.

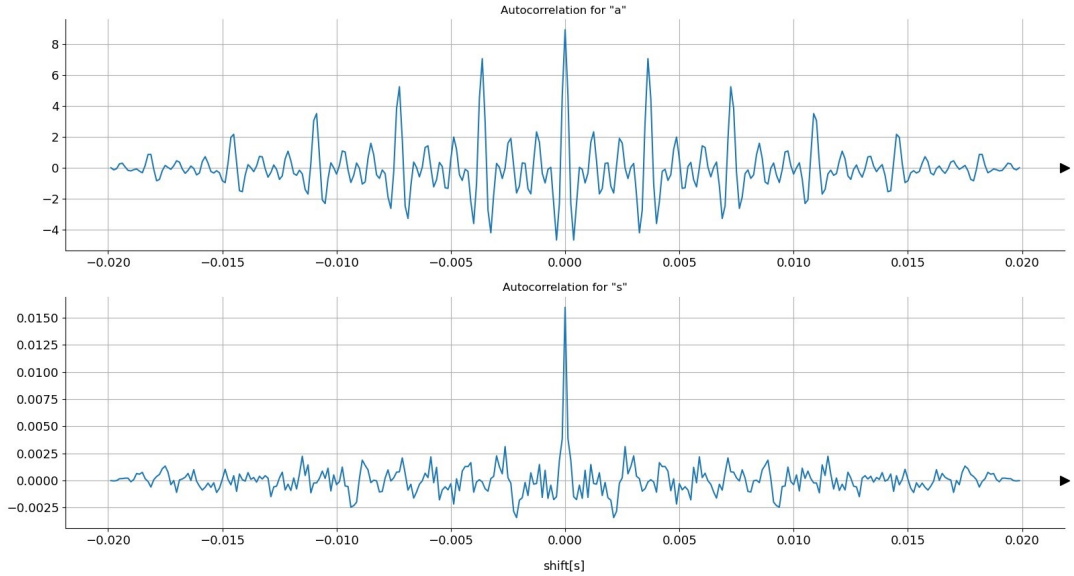


Figure 1.22: Autocorrelation of the *'a'* and *'s'* phonemes

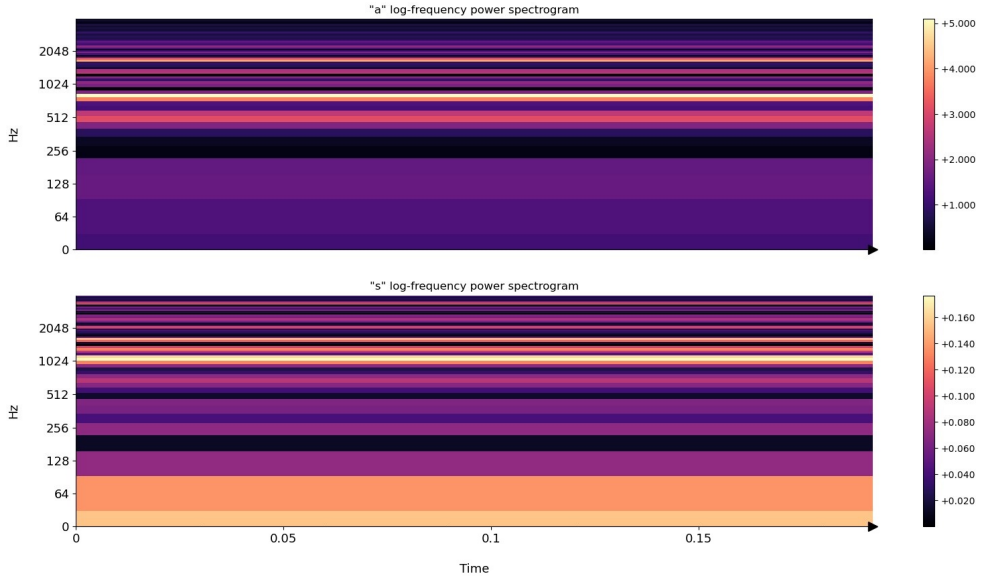


Figure 1.23: Spectrograms of the *'a'* and *'s'* phonemes

1.3.2 Effects of Pathologies on the Vocal Signal

In order to exhibit an effect on the vocal signal, a pathology must influence one or more of the elements responsible for the speech process, i.e. the respiratory system, the brain or the vocal tract. Multiple such pathologies exhibit unique characteristics that distinctly alter the nature of the vocal signal, either by the effect on the sound quality or intensity or by the effect on the coordination of the articulators [44].

1.3.3 Examples of Conditions that affect the Human Speech

Vocal Cord Cysts

The voices of people ailed by vocal cord cysts may sound hoarse, breathy or dry. Other symptoms might include: vocal fatigue, multiple tones, vocal range loss or voice loss. Vocal cord cysts are formations comprised of a sac formed around a semisolid or fluid-filled center [45, 46]. This pathology

will be further referred to as ”Cysts”.

Ventricular Dysphonia

Ventricular Dysphonia also known for its latin name, *Dysphonia plicarum ventricularum*, is a disorder that affects speech by pathologically engaging the ventricular folds (also known as false vocal chords) in the process of phonation instead of the vocal folds. A voice affected by this pathology is described as hoarse and stifled with a lower fundamental tone [47].

The term pocket fold voice as an expression of hyperkinetic dysphonia refers to a changed articulation of the human voice due to the extreme compression of the pocket folds. The symptoms are a very hoarse, strained voice.

Parkinson's Disease

The voices of people suffering from Parkinson's Disease are described as ”soft, monotone and hoarse” with ”uncertain articulation”. People ailed by Parkinson's Disease also tend to stutter or mumble during sentences or their voices might trail off at the end of sentences due to the effects of the disease on the motor and cognitive systems [48, 49].

1.3.4 Speech Signal Recording Methods

The following two methods were used to obtain the signals analyzed in this work:

- **Sound Wave Recordings of the Vocal Signal**

  Speech is usually recorded with a microphone that translates the vibration of air particles to an alternative voltage or current with a variable capacitor or inductor. The quality of a vocal recordings' database will be affected by the types of microphones, the sampling rate used in the recording process and by the recording environment's properties.

- **Electroglottography**

  Electroglottography is a method that uses the electroglottograph to monitor the vocal folds' vibrations. The electroglottograph is a non-invasive device that measures variation of the neck tissues' impedance to a weak alternating current. The impedance varies in conjunction with the measured subject's speech [50].

1.3.5 Neural Networks and Classification

A classifier (or discriminator) is defined as an algorithm capable to categorize the datapoints from the input, denoted by X, into two (in the instance of a two class discriminator) or more (in the instance of a multi-class discriminator) classes, denoted by Y. Usually, a complicated function, without a known mathematical expression is required to solve a classification problem, by mapping an input, $x^{(i)}$, to the desired output, $y^{(j)}$. Therefore, a system capable to automatically learn the function is required.

Such an algorithm can be represented by a neural network that consists of a set of elementery units, linked by connections of different weights, called (artificial) neurons. The neurons, named after their similarity to the biological neurons in the brain, are hierarchically organized in layers. If every neuron from a given layer $[l]$ receives information from all neurons in the previous layer, $[l-1]$, the neural network is also known as an (FCNN) or fully connected neural network [51].

The main characteristics of an FCNN are called hyper-parameters that are enumerated below:

- the number of layers
- the number of neurons in each layer

- the activation function of each layer
- the optimizer
- the learning rate
- the number of epochs
- the batch size

The **number of layers** represents the total number of sets of neurons. The three categories of layers in a neural network are: input layer, that consists of a number of neurons equal to the dimension of an input datapoint, $x^{(i)}$ (i.e. features); hidden layers, which can have any dimension (i.e. number of neurons), and the output layer that has a dimension equal to the number of classes required to solve the classification problem.

Fully Connected or Dense layers

The **number of neurons** in each layer is equivalent to the number of units in that layer. One unit consists of two components: a weight and a bias component that are iteratively calculated by the algorithm itself.

Given a neuron from a hidden layer, l, the following equation denotes its linear part:

$$z = b + \sum_{i=0}^{n^{[l-1]}-1} a^{(i)} w^{(i)}, \tag{1.47}$$

where $n^{[l-1]}$ represents the number of outputs that originate from the previous hidden layer, $l-1$, $a^{(i)}$ is the input from i^{th} neuron of the previous layer, $w^{(i)}$ is the weight that connects the i^{th} neuron to the current neuron and b is the bias term of the current neuron from, l. Figure 1.24 depicts two consecutive Dense layers. Each neuron from the second layer receives data from all the neurons in the previous layer to exemplify a section of a fully connected network.

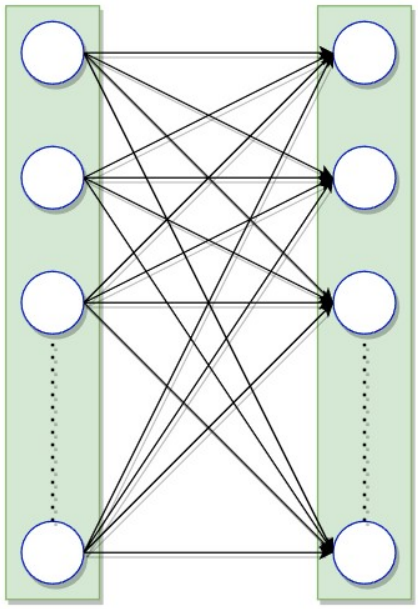


Figure 1.24: Dense Layer Example

Convolutional layers

A **convolutional layer** is a type of layer in an artificial neural network that performs convolution on its input. The most common application of a convolutional layer is represented by feature extraction. To perform feature extraction, a convolutional layer uses an n dimensional filter (also known as kernel) of learnable weights that matches the dimensions of the input such that convolution is possible between the input and the filter. The size of a filter is smaller than the size of the input and the multiplication from the convolution operation is performed on parts of the input signal that match the kernal's size. By convention, multiplication is repeated in the ascending order of each dimension's coordinates. In the case of a bidimensional example, convolution is performed from left to right and from top to bottom.

The stride is the number of datapoints by which a kernel moves to calculate the convolution on the next part of the signal. Padding may be applied around the input datapoints to ensure a desired output shape and that the filter remains within bounds of the input tensor for a given stride. Padding represents the completion of the input tensor with values around its bounds. Figure 1.25 exemplifies the convolution between a square padded matrix of 10×10 dimensions. The kernel is a 3×3 square and the output value that is currently calculated was highlighted with light green.

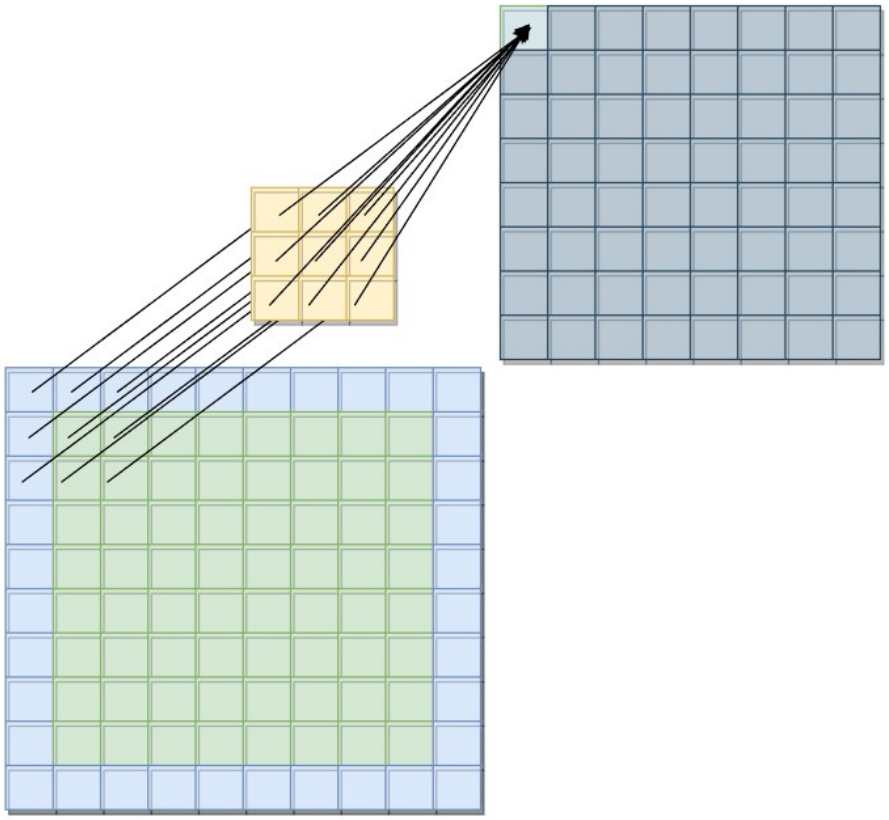


Figure 1.25: Convolutional Layer Example

Types of Activation Functions

The linear output z is passed forward as the input of the **activation function** that has the role to introduce a non-linearity, because most classification problems require mapping an intrinsically non-linear relationship between the inputs and outputs. The most used activation functions ($g(x)$) for the hidden layers are:

- The *ReLU* function, which has the following mathematical formula [52]:

$$g(x) = max(0, x) \tag{1.48}$$

- The leaky *GeLU* function [52]:

$$g(x) = max(\alpha x, x), \tag{1.49}$$

where $\alpha \in (0, 1)$

- The *ELU* function [52]:

$$g(x) = max(\alpha(e^x - 1), x), \tag{1.50}$$

where $\alpha \in (0, 1)$

- The *Sigmoid* function [52]:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{1.51}$$

- The *Hyperbolic Tangent* function [52]:

$$g(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.52}$$

Optimizers

An **optimizer** is an algorithm that seeks a minimum value in a mathematical optimization problem of a function that usually depends on a large number of parameters. The iterative update of a neural network's parameters(that are also known as weights) can be viewed as such a problem. The optimization problem described as an iterative update, in the hyperspace of the neural network's parameters, with the goal to reduce the value of a loss function that essentially denotes the error between the model's output and the ideal(desired) output. The learning process is represented by the iterative update of the weights of the neural network.

The **batch size** represents the number of adjacent examples from the training dataset that is fed to the neural network at a certain moment (i.e. iteration).

An **epoch** represents the series of iterations that imply a complete pass through the training set of data.

The dimension and direction of the update step taken by the optimizer at each iteration is determined by the **learning rate** hyperparameter. Some optimizers have an adaptive learning rate, for example the ADAM optimizer [53].

Overfitting and Underfitting

To asses the quality of the classification achieved by the neural network, the corpus of data must be split into three distinct sets: training, validation and test sets. The training set contains the data from which the neural network learns. The validation set contains the data used to monitor the accuracy improvements during the learning process. The optimizer does not update the weights' values based on the validation loss (i.e. the loss function calculated for the validation data), but only based on the value of the training loss. Finally, the test set is a set of data (distinct from the training and validation sets) that is used to asses the performance of the classification model on real-life data [54, 55].

Overfitting means that the model has learned the training data too closely, such that the analysis may not be reliably generalized on future predictions. Overfitting can be observed when the training loss is low and the training accuracy is high, while the validation and test accuracies are low and their corresponding loss functions have high values.

Underfitting refers to the situation where a model is unable to understand the relationship between the input and the associated output.

The Bias and Variance trade-off

To understand how the function approximated by the model deviates from the real function that maps the input to the data, bias and variance can be used to illustrate the underlying phenomena. Bias represents the simplifying assumptions made by the model for making easier approximations of the target function. Variance is the fluctuation of the function estimated by the model over different training examples.

The relationship between bias and variance is related to the problems of overfitting, underfitting and generalization capacity in machine learning. More precisely, when the generalization error is calculated (where bias and variance represent essential factors), an increase in the model's generalization capacity can result in an increase of the variance and decrease in bias. Therefore, high variance can be associated with overfitting and high bias can be associated with underfitting.

Hence, a data model with a good generalization capacity requires a low bias when modelling data with a high accuracy. Additionally, the results obtained from the output of the model should not be excessively varied and therefore, for good performance of the model, a low variance is useful.

One common way to reduce overfitting is with the use of regularization. **Regularization** is a technique that reduces overfitting by reducing some of the weights to encourage a higher flexibility of the model [56].

Most Common types of Regularization

Both $L1$ and $L2$ regularizers apply a penalty term to the loss function (also known as cost function). The penalty term is proportional to the size of the neural network and affects only the weight terms. The main difference is the formula for the term of each regularizer [57].

The L_1 regularization

The expression for the L_1 regularization term that will be added to the loss function of a neural network is:

$$L1 = \frac{\lambda}{n} \sum_{l=0}^{L-1} \left(\sum_{i=0}^{n^{[l-1]}-1} \sum_{j=0}^{n^{[l]}-1} |w_{i,j}^{[l]}| \right), \tag{1.53}$$

where L is the number of layers in the neural network, $n^{[l]}$ is the input size of the l^{th} layer and λ is called the regularization parameter. Using L_1 regularization yields many values of zero in the **w** vector making the weight vector sparse.

The L_2 regularization

The expression for the L_2 regularization term that will be added to the loss function of a neural network is:

$$L_2 = \frac{\lambda}{2n} \sum_{l=0}^{L-1} \left(\sum_{i=0}^{n^{[l-1]}-1} \sum_{j=0}^{n^{[l]}-1} |w_{i,j}^{[l]}|^2 \right) \tag{1.54}$$

If the chosen value of the λ hyperparameter is too low, the effect of the regularization will insufficiently reduce variance, hence will fail to counter the model's overfitting. Conversely, if the value of λ is too high, the regularization will increase the bias and therefore will result in underfitting.

Dropout The **dropout** regularization method consists in randomly ignoring a number of layer outputs that are essentially "dropped out". In practice, this means that, for a percentage of neurons, all their output weights are set to zero, so their contribution to the input of the further layers is ignored [58, 59].

Chapter 2
Automatic Voice Pathology Detection Algorithm

2.1 Problem Formulation

The general outline of the automatic diagnosis algorithm is shown in Figure 2.1. The input considered consists of 1s audio or EGG signals of the voice of different speakers. In the case of two channel input signals, the signals were converted to single channel and then further divided into 20ms overlapping windows. The resulting windows from a given vocal recording will be further referred to as the windowed signal. Assuming the input signal was not analysed yet, the windowed signal is then saved to the hard drive to accelerate further executions of the process. From each window of a windowed signal, the system automatically extracts a series of features in different domains (e.g. time domain, frequency domain, cepstral domain, etc.). The resulting tensor will be essentially a list of either w feature vectors or matrices, where w is the number of windows in a windowed signal. The extraction process concludes with the storage of the feature tensors to the hard drive. A significant obstacle in finding a general solution for the problem is given by the presence of multiple pathologies in some speakers of the available datasets.

The storage phases were introduced to avoid signal windowing and feature extraction at each execution of the algorithm since windowing and feature extraction are the longest lasting preprocessing stages.

The aforementioned feature tensors are forwarded to a classifier based on a neural network that will output a probability associated with each disease for every input tensor.

The proposed system is designed in a modular and flexible manner, allowing for parts of the pipeline to be adapted or changed in accordance with the application domain. The feature extractor allows the user to implement or import diverse extraction functions adapted to the problem and to choose which functions will be used for the extraction of the compound feature tensors. Additionally, the proposed design allows the classification of data labeled with an arbitrarily large number of classes.

2.2 Experimental Setup

Figure 2.2 depicts the complete structure of the Audio Disease Classification algorithm. Every stage of the figure will be elaborated in the following sections. To implement the automatic voice pathology detection algorithm, the Python programming language (version 3.7.4) [60] was used along with the following libraries and Application Programming Interfaces (APIs): Tensorflow 2.2 [61], Librosa

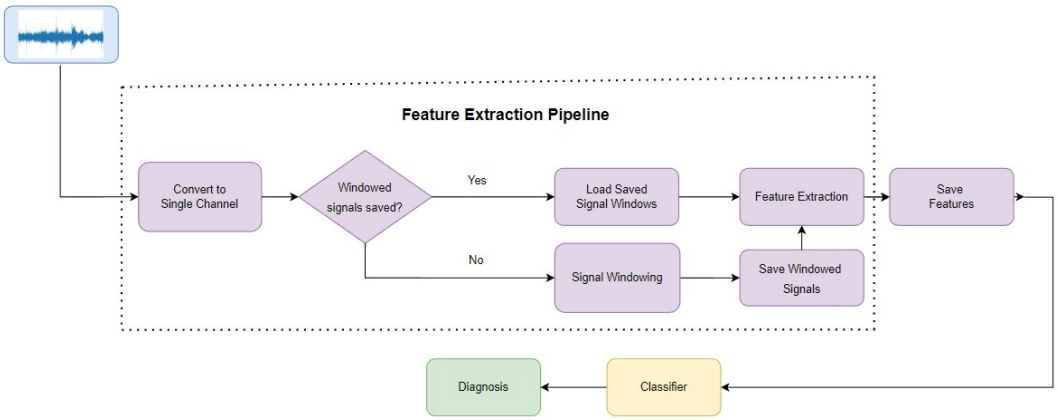


Figure 2.1: General Overview of the Classification Process

[62], NumPy [63], Scipy [64], Scikit-Learn [65]. The machine used to develop the proposed system is a Lenovo Legion Y7000 laptop with the following configuration:

- Nvidia GeForce GTX 1650 GPU
- 9^{th} generation Intel Core i7 CPU
- 24 GB RAM memory

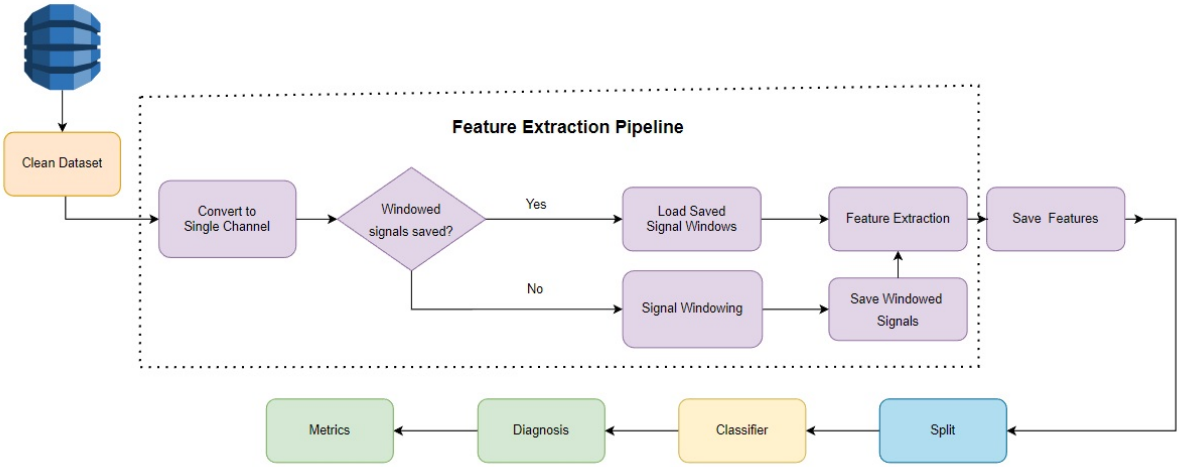


Figure 2.2: Complete Overview of the Algorithm Implementation

2.2.1 Dataset

The Saarbrucken Voice Dataset (SVD) is a dataset of audio and electroglottograph (EGG) recordings of 869 healthy and 1356 speakers (candidates) with one or more of 71 voice disorders sampled at 50kHz [66, 2]. Each candidate has an associated speaker number. For each speaker, there are 10 vocal recordings and 10 EGG recordings: 9 recordings of the vowels /i/, /a/, /u/ pronounced in three different intonations (low, normal and high) and one recording of the sentence ”Good morning, how are you?” uttered in German. Each recording of the vowels is labeled suggestively using the following file name

pattern: $speakerID - vowel\_intonation(-egg).ext$, where "-egg" is optional, $vowel \in \{'i', 'a', 'u'\}$, $intonation \in \{'l', 'n', 'h'\}$, where 'l', 'n', 'h' stand for low, normal, high respectively and $ext \in \{'.wav', '.egg'\}$, the extension of the file. For example, $1 - a\_l.wav$ is the audio recording of vowel 'a' pronounced with low intonation by the 1^{st} speaker. The dataset also includes information about the age and gender of the subjects alongside the clinical diagnosis. The dataset can be downloaded on the official site of the University of Saarland from Germany.

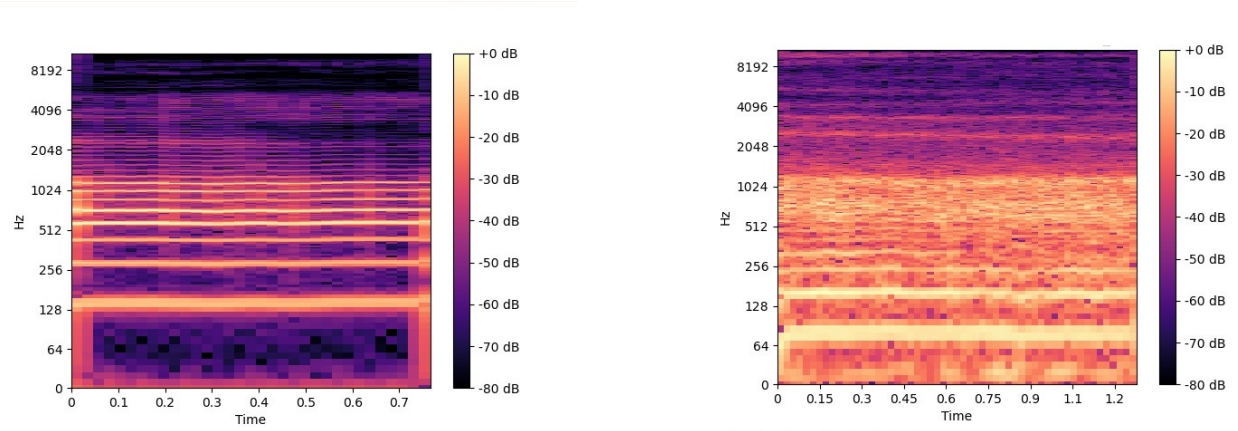
For the experiments in this work, an excerpt of three diagnostics was used: Healthy ("gesund" in German), Cysts ("Cyste" in German) and Ventricular Dysphonia ("Taschenfaltenstimme" in German). For an easier separation, the three diagnostics were downloaded in three distinct archives. Each archive was extracted to the folder named after the corresponding diagnostic. Each example from the dataset was labeled based on the containing folder. For instance, if signal $1 - a\_l.wav$ is found in the folder "Healthy", the label of the concerned signal is "Healthy". The pathologies in the excerpt were chosen such that the speakers do not have true labels of more than one diagnostic (i.e. speaker X will not be present in both the "Cysts" and "Ventricular Dysphonia" folders).

Table 2.1 shows the gender distribution from the selected excerpt of the SVD. The excerpt was chosen such that the candidates with one pathology did not suffer from other pathologies in the excerpt.

	Healthy	Cysts	Ventricular Dysphonia
Male	433	1	10
Female	436	5	1
Total	869	6	11

Table 2.1: Gender Distribution of the Dataset Excerpt, Source: [2]

Figure 2.3 depicts the spectral differences between the recordings of a healthy person and a person suffering from ventricular dysphonia from the SVD. The comparison was made on the 'high' intonation of vowel 'a'.



(a) Spectrogram of a Healthy Subject

(b) Spectrogram for Ventricular Dysphonia

Figure 2.3: Spectrogram Comparison for Diagnostics

Dataset preparation

Due to the highly imbalanced nature of the data from the excerpt of the SVD, the `clean_dataset.py` script was implemented. Based on the pattern of the recordings' names, the script allows the user to: balance the dataset so that each diagnostic will contain an equal number of speakers, delete the recordings below a given duration, keep only a desired set of vowels and intonations from the excerpt of the dataset that will be further referred to as "the dataset".

2.2.2 Preprocessing and Feature Extraction

The system can load both audio and EGG recordings from the dataset, using the `librosa.load()` function, depending on the user's choice. The feature extraction pipeline begins with the discovery of the shortest signal length from the dataset, denoted by l_{min}. This step was necessary to ensure an equal number of signal windows for each class, because the recordings varied in length by more than 20ms.

Conversion to single channel

If the windowed signals are not to be found on the hard drive of the machine that executes the algorithm, the next step in the feature extraction pipeline was the conversion of every multi-channel signal to a single channel signal by averaging the signal across all channels as follows:

$$x(n) = x_{mono}(n) = \frac{1}{n_{ch}} \sum_{k=0}^{n_{ch}-1} x_{multi}^{(k)}(n), \tag{2.1}$$

where $x_{multi}^{(k)}$ is the k^{th} channel of the multichannel signal, x_{multi} and n_{ch} is the number of channels of the multichannel signal.

Signal Windowing

The converted signals were then trimmed to the length of the shortest signal from the dataset and their trimmed versions are windowed with user-specified fixed length windows that overlap by a user-specified percentage (denoted by ov).

$$\begin{aligned} w_k(n) = x(n)w_D(n - ks), \quad s = round((1 - ov)D), \\ n \in \{0, 1, ..., l_{min} - 1\}, \quad k \in \{0, 1, ..., n_{win}\}, ov \in [0, 1), \end{aligned} \tag{2.2}$$

where *nwin* is the number of windows, D is the length of a window, s is the step and n_{win} is calculated by[67]:

$$n_{win} = \frac{l_{min} - D}{1 - ov} + 1 \tag{2.3}$$

The windowing function also allows the user to choose the window types used for signal windowing from the `scipy.signal.windows` package.

Saving the windows

The windowed signals were saved with the `.npy` extension, denoting the standard binary format used by the built in Python library, `numpy` for persistent storage of a tensor data structure called NumPy Array in a `.npy` file on the disk. The reason for this operation is the goal of a flexible algorithm with partially independent stages. Therefore, the training stage of the task concerned will not depend heavily on the initial windowing function as will be revealed in the further steps (if the signals have already been windowed).

Implemented Types of Feature Extractors In the context of the proposed system, a feature extractor is represented by a function that processes each window. A feature extractor can represent the input signal as either a scalar (e.g. the energy of a given signal), a vector (e.g. the amplitude spectrum of the signal) or a matrix (e.g. the spectrogram of the signal). The feature tensor of a voice record is obtained by concatenating the outputs of a series of user-specified feature extractors with user-specified parameters. If the feature tensors contain features extracted by multiple extractors, these tensors will be referred to as "compound features" or "compound tensors". The user can choose

whether a feature tensor will be further compressed by extracting the mean and a type of variance from each feature of the compound tensor. The two variance types implemented in the proposed solution are variance and squared median absolute deviation (SMAD) and are defined in the following equations:

$$\sigma^2 = \frac{\sum_{i=0}^{N-1}(x_i - \mu)^2}{N-1}, \tag{2.4}$$

where μ represents the mean of the vector $(x)_{(1\leq i\leq N)}$.

$$\sigma^2 = (median(|x_i - \mu|))^2, \tag{2.5}$$

where μ represents the mean of the vector $(x)_{(1\leq i\leq N)}$.

An uncompressed feature tensor will be referred to as "raw features" or "raw feature tensor". The following selection of feature extractors was used in the experiments from the following chapter:

- DFT

  The Discrete Fourier Transform has been implemented using the Fast Fourier Transform algorithm from the NumPy library. This implementation is a vectorized version of the Fast Fourier Transform meaning that the function can compute the DFT on a specified dimension of the input tensor. The input tensor is represented by a matrix of windows in the case of the proposed system. The user can specify the number of points for the DFT computation via the N_{fft} parameter. The DFT function returns the amplitude spectrum of the input tensor.

- STFT

  The STFT function implements the `STFT` function from the librosa package and computes the amplitude of the STFT. The user can specify the following parameters: N_{fft} and hop_len. The latter parameter refers to the size of the hop defined in the STFT equation 1.34 from the previous chapter.

- Mel-spectrogram

  The Mel-spectrogram function implements the `melspectrogram` from librosa and depends on the same user-specified parameters as the STFT function.

- Cepstrum

  The Cepstrum function implements the NumPy versions of the DFT and logarithm functions as defined by the equation 1.20 in the previous chapter and the base of the logarithm has a fixed value of 10. The Cepstrum function depends on the same user-specified parameters as the DFT fuction.

- MFCC

  The MFCC function implements the `MFCC` from librosa and depends on the same user-specified parameters as the STFT function and an additional n_{MFCC} denoting the number of bins on the Mel scale as defined by equation 1.43 in the previous chapter.

Saving the Features

The extracted features were saved to ensure a higher flexibility of the algorithm. This allows a training process independent of the feature extraction and offers the user the capacity to experiment with different neural network architectures without repeating the extraction process for every training experiment. Saving the data is especially useful when multiple training experiments are run and the

extraction process is time consuming.

Test-Train Split

After the features are saved, the `split_dataset.py` script splits the dataset, specified by the `split_path` parameter, into training and testing subsets with the following user-specified parameters: `split_perc`, `perspeaker`. Parameter `split_perc` specifies what percentage of the whole dataset will be represented by the training set, while the rest of $(100 - split\_perc)\%$ will be represented by the test set. Parameter `perspeaker` is a boolean for the choice of splitting the dataset: intra-speaker or inter-speaker. If the value is set to `True`, the dataset will be split such that all the recordings of `split_perc` speakers will be found in the training set, while the testing set will be constituted of other speakers with different vocal parameters than the speakers of the training set. If the parameter is set to false, the training and test sets will likely contain recordings from the same speakers, therefore the datasets will be shuffled at recording level and not speaker level. "Speaker level" (also referred to as "per speaker") refers to multiple recordings (up to 9), depending on what vowels and intonations were chosen by the user in the dataset preparation stage.

2.2.3 Neural Network Based Classifier

A neural network based classifier best operates with scaled data to avoid the issue of exploding gradients. The exploding gradients problem represents a common issue in machine learning, because the magnitude of the gradients, used to optimize the weights in gradient-based optimization methods, varies exponentially with the depth (i.e. number of layers) of the network, hence the values of the gradients "explode". Therefore, the training data is scaled befeore being used for training.

Data scaling and Scaler types

The most common data scaling intervals used in machine learning are $[0, 1]$ and $[-1, 1]$ [68]. A set of datapoints, $s(n)$ is scaled to [0,1] when applied the following operations:

$$s_{scaled}(n) = \frac{s(n) + |min(s(n))|}{|min(s(n))| + |max(s(n))|}, \tag{2.6}$$

where max(s(n)) and min(s(n)) are the minimum and maximum values from the given set of datapoints. This type of scaling is also known as Minimum-Maximum(or MinMax) Scaling.

A set of datapoints, $s(n)$ is scaled to [-1,1] when applied the following operations:

$$s_{scaled}(n) = \frac{s(n)}{max(|s(n)|)} \tag{2.7}$$

This type of scaling is also known as Maximum Absolute(or MaxAbs) Scaling.

In the data scaling process, one of six scaler classes can be used depeding on the shape of the data: two of them imported from the Scikit-Learn (or sklearn for short) library and the other four were custom written MinMax and MaxAbs scalers.

The scalers from sklearn can be used if the data tensors' dimensions are exactly 1 (i.e. if the input data is represented by a list of compound feature vectors (i.e. a matrix), where each column represents a feature.

The scalers from sklearn will compute the minimum and maximum values (in case of the MinMax scaler) or the maximum absolute value (in case of the MaxAbs scaler) for every column in the matrix. A set of data that had its statistical parameters (i.e. minimum, maximum, mean, etc.) priorly determined will be referred to as "fit data". The fit data can then be scaled columnwise.

The other four scalers have been created to allow global scaling or scaling over different, user-specified dimensions for two-dimensional data tensors (i.e. matrices instead of vectors). Global

scaling refers to scaling with statistical parameters that are calculated for the whole list of data tensors instead of a given dimension.

K-fold Cross-Validation and Training process parameters

The scaled data is then passed to a k-fold cross-validation algorithm that further splits the training data into training and validation sets. The `k_fold_cross_validation` function allows the user to specify the parameters of the training process:

- k – the number of subsets to split the training data into
- batch_size – integer specifying the number of data tensors passed to the neural network in one training iteration
- shuffle_buffer – integer specifying the size of the buffer used to shuffle the data
- shuffle_mode – boolean specifying whether the test data and the validation data should be shuffled as one dataset
- epochs – integer specifying the number of epochs to train a given model
- optimizer – string specifying the name of the optimizer choice from the `tensorflow.keras.optimizers` package
- dropout – floating point number in [0,1] denoting the percentage of neurons to be affected by Dropout regularization

The k-fold cross-validation algorithm is a method used to prevent overfitting. The algoritm consists in splitting the training data into k subgroups and train k classification models, where each classifier's validation data is represented by a different subgroup and the training data will be represented by the other k-1 subgroups as 2.4 shows [69].

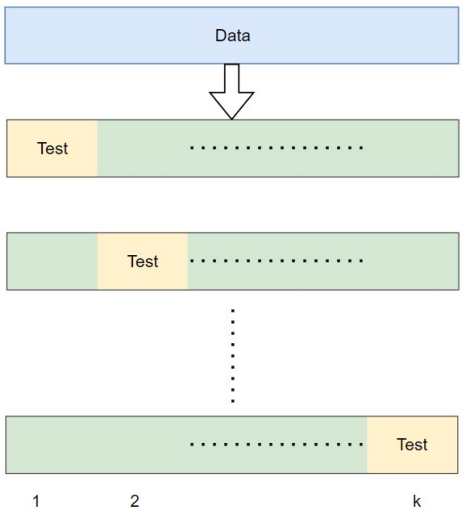


Figure 2.4: K-fold Cross-Validation example

The Loss function minimized in the training process Given the nature of the classification problem, categorical crossentropy (CCE) was chosen as the loss function between the outputs, $\hat{y}^{(i)}$ estimated by the network and the real labels $y^{(i)}$ because classification can be intuitively viewed as associating the most likely class, $\hat{y}^{(i)}$ to the corresponding data tensor, $x^{(i)}$. The estimations of the network were provided by the softmax activation function from the output fully connected layer.

The expression of the **categorical cross-entropy** is defined in the following equation:

$$CE = -\sum_{i=1}^{C} s_i log(\hat{s}_i), \tag{2.8}$$

where C is the number of classes, $\hat{s}_i$ is the score of the predicted class and s_i is the score of the i_{th} class. In essence, the CCE calculates a distance between two PMFs

The Softmax activation function's purpose is to transform the vector of **z** into a PMF that allows the CCE loss function to be computed. The expression of the Softmax function is defined in the following equation:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \tag{2.9}$$

where C is the number of classes.

Initially, the labels of the data are represented by indices of a class list, where each index, $i \in \{0, 1, ..., C-1\}$ and C is the number of classes. To minimize the loss function, the format of the real labels, $y^{(i)}$, was first modified to fit the pattern of a PMF where the value of the correct class index was 1 and the values of the other indices were 0 so that te shapes of the real labels, $y^{(i)}$, would match the shape of the estimations, $\hat{y}^{(i)}$.

The optimizer chosen for the training process

To minimize the CCE, the ADAM [53] optimizer was used. The name ADAM is inspired from the term Adaptive Moment Estimation. The algorithm was proposed as an adaptive gradient-based optimization method with a faster convergence than the Steepest Descent method.

Gradient Descent (also known as Steepest Descent) is an example for gradient-based optimizers and consists of an iterative uptade of the weights of a neural network with a step proportional to the gradient of the highest magnitude. An iteration of the Gradient Descent algorithm is given by the following formula:

$$\mathbf{w_{i+1}} = \mathbf{w_i} - \alpha \frac{\partial}{\partial \mathbf{w_i}} L(\mathbf{X}, \mathbf{w_i}), \tag{2.10}$$

where $\mathbf{w_i}$ is the value of the weight vector, $\mathbf{w}$ at the i^{th} iterationm α is the learning rate and $L(\mathbf{w})$ is the loss function calculated at the i^{th} iteration.

The ADAM optimizer seeks to find individual learning rates for each parameter. This optimizationmethod scales the learning rate by some factor of the estimations of the first and second statistical moments of the gradient of a function. The N^{th} statistical moment, $m^{(N)}$ of a random variable X is defined by the following expression:

$$m^{(N)} = E(X^N) = \sum_{x=-\infty}^{\infty} x^N P_X(x), \tag{2.11}$$

where $E(X^N)$ is the expected value of the variable X^N. The gradient is considered a random variable because the analytical expression of the loss function $L(\mathbf{X}, \mathbf{w})$ is unknown with respect to the variable $\mathbf{w}$. The estimations of the 1^{st} and 2^{nd} moments of the gradient are evaluated using exponentially weighted moving averages for the i^{th} iteration that corresponds to the i^{th} mini-batch of data with the following formulae:

$$m_i^{(1)} = \beta_1 m_{i-1} + (1 - \beta_1) g_i \tag{2.12}$$

$$m_i^{(2)} = \beta_2 v_{i-1} + (1-\beta_2) g_i^2, \tag{2.13}$$

where:

- $m_i^{(1)}$ – 1^{st} moment estimate of the gradient
- $m_i^{(2)}$ – 2^{nd} moment estimate of the gradient
- β_1 and β_2 are the exponential decay rates of the moments with usual values of 0.9 and 0.999 respectively
- $g_i = \frac{\partial}{\partial \mathbf{w_i}} L(\mathbf{X}, \mathbf{w_i})$ – gradient of the loss function at iteration i

Because the initial values of the 1^{st} and 2^{nd} moment estimators, $m_0^{(1)}$ and $m_0^{(2)}$ are set to 0, the estimators' values will be biased toward 0. The bias corrected versions of the two moments are defined in the following equations: $\hat{y}^{(i)}$

$$\hat{m}_i^{(1)} = \frac{m_i^{(1)}}{1-\beta_1} \tag{2.14}$$

$$\hat{m}_i^{(2)} = \frac{m_i^{(2)}}{1-\beta_2} \tag{2.15}$$

Hence, the update of the ADAM optimizer will have the expression:

$$\mathbf{w_{i+1}} = \mathbf{w_i} - \alpha \frac{\hat{m}_i^{(1)}}{\sqrt{\hat{m}_i^{(2)} + \epsilon}} g_i, \tag{2.16}$$

where ϵ is a small value chosen to avoid division by 0, usually 10^{-10}.

Metrics Used to Measure the Efficiency of the Proposed System

The Confusion Matrix

The confusion matrix, $CM[i][j]$, is a tabular metric used to describe the performance of a C class classifier on a test set, where the row indices $i \in \{0, 1, ..., C-1\}$ specify the true (expected) label, y, of the estimation, $\hat{y}$ and the column indices, $j \in \{0, 1, ..., C-1\}$ specify the predicted label, $\hat{y}$. The value of $CM[i][j]$ indicates how many examples of the i^{th} class have been classified as the j^{th} class. Figure 2.5a depicts an example confusion matrix where 211 examples of Class 2 have been misclassified as Class 1:

The Accuracy an Per-Class Accuracy The overall accuracy of a classifier can be defined from its confusion matrix with the formula:

$$acc = \frac{\sum_{k=0}^{C-1} CM[k][k]}{\sum_{i=0}^{C-1} \sum_{j=0}^{C-1} CM[i][j]} \tag{2.17}$$

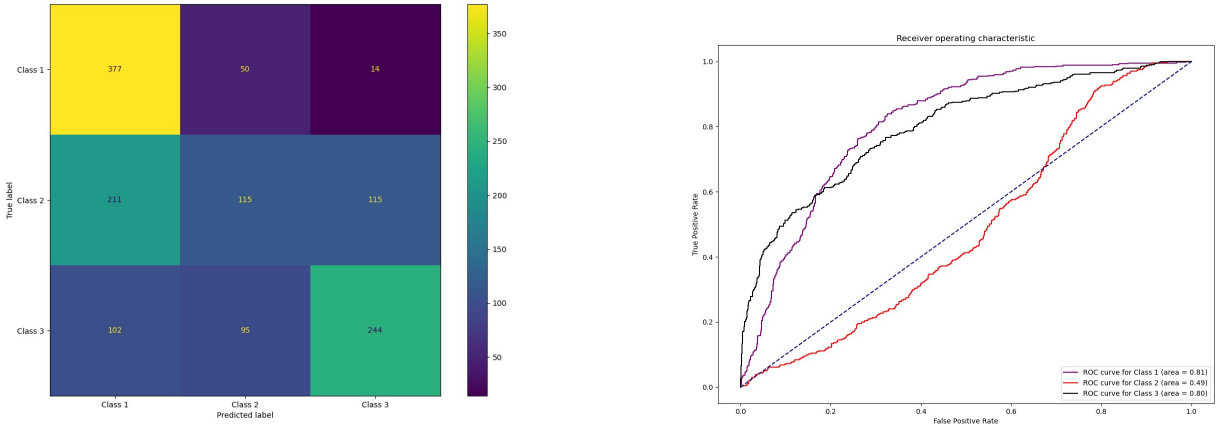
The per-class accuracy, acc_k , of the class k is defined by the following equation:

$$acc_k = \frac{CM[k][k]}{\sum_{i=0}^{C-1} CM[i][k]}, \tag{2.18}$$

where C is the number of classes.

The AUC-ROC metric

The Receiver Characteristic Operator (ROC) is a metric for the performance evaluation of a classifier, usually in problems of binary classification. The is defined as the rate true positives (i.e. percentage of the correctly identified positive class) compared to the rate of false positives ROC (i.e. percentage of negative class misclassified as the positive class) as the discrimination threshold varies. The discrimination threshold can be viewed as a hyperplane in the hyperspace of the neural network's parameters. In case of classification problems with multiple classes, this method is regarded as a One-vs-Rest approach. The capability of the model to correctly identify classes is measured by the Area Under Curve (AUC) score. Figure 2.5b depicts an example for the AUC-ROC metric.



(a) Confusion Matrix Example

(b) AUC-ROC Example

Figure 2.5: Example Metrics

Chapter 3
Experiments and Results

3.1 Preprocessing and Feature Extraction Settings

3.1.1 Dataset preparation

For the training process, the recordings of some speakers in the Healthy and Ventricular Dysphonia folders were discarded from both the audio and EGG datasets to ensure balanced classes for the classification problem. The resulting datasets consist of three pathologies with 6 speakers for each diagnostic and for each speaker, there are 9 recordings of the vowels: 'i', 'a', 'u', each pronounced with all of the following intonations: low, normal and high.

3.1.2 Windowing:

All the recordings from the datasets were trimmed to the length of the shortest recording in the corresponding dataset and were windowed as means of data augmentation. Data augmentation is a technique to increase the amount of training data by either addition of newly generated, synthetic data or by addition of slightly modified copies of the examples in the dataset. The goal of data augmentation is to reduce overfitting by increasing the variability of a small collection [70].

The following settings were used for the windowing function throughout all the experiments:

- window type: rectangular
- window length: 20 milliseconds to ensure quasi-stationarity of the vocal recordings [71]
- overlap: 50%

3.1.3 Feature Extractor Class

For both datasets, the features were raw MFCCs, extracted with the following settings:

- $N_{fft} = 2048$
- $hop\_len = 128$
- $n_{MFCC} = 26$

The extraction resulted in data tensors represented by matrices of 8 time bins on the horizontal axis, each containing 26 MFCCs on the vertical axis. For ease of notation, the matrices can be considered of shape $(26, 8)$, where shape (r, c) denotes the number of rows and columns as r and c respectively.

3.1.4 Test-Train Split

The dataset was split on a speaker level (i.e. the perspeaker was set to True) to ensure that the model learns the characteristics of a pathology instead of the characteristics of a given speaker. The split_perc parameter was set to 84% to split the dataset into: 5 speakers per training set and 1 speaker per test set.

3.1.5 Data Scaling

The data was scaled using the global MaxAbs custom scaler that calculates a global absolute maximum.

3.1.6 Classifier Settings and Used Architecture

The proposed neural architecture used in all the experiments is shown in figure 3.1. The convolutional layer can be viewed as an additional feature extractor and was used to improve the generalization capability of the classifier, while reducing its complexity, as the authors of [72] and [73] demonstrated.

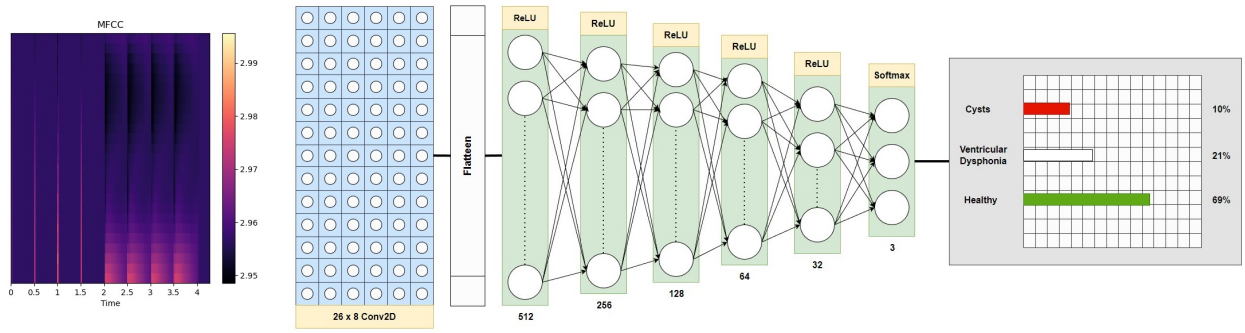


Figure 3.1: Proposed Architecture

After the convolutional layer, no pooling layer was used because downsampling the data was not required. The ReLU function was used as the activation for every hidden layer because of the short computation time (compared to other commonly used activation functions, such as hyperbolic tangent or sigmoid functions) and does not suffer from the problem of vanishing gradients (i.e. the gradients do not tend to exponentially decrease with the depth of the network), hence the learning process is faster. Batch normalization was performed after the first Dense layer after the activation of the first Dense layer, since the convolutional layer can be regarded as part of the feature extractor. Dropout was used as a regularization method for all the hidden Dense layers with a dropout rate of 0.5.

The reason for using MFCCs extracted on sub windows consists in repeated training attempts which shown that the extraction of MFCCs on 8 sub-windows from each signal window yielded significantly better results in terms of accuracy than the MFCC extraction from the entire window. A possible explanation for the utility of the presented approach is that sub-windowing better captures pathology specific time dependencies.

3.1.7 Experiments and Performance Evaluation

Audio Signal Dataset

Figures 3.2 and 3.3 depict the evolution of the training and validation losses and accuracies respectively for 130 epochs on the audio dataset:

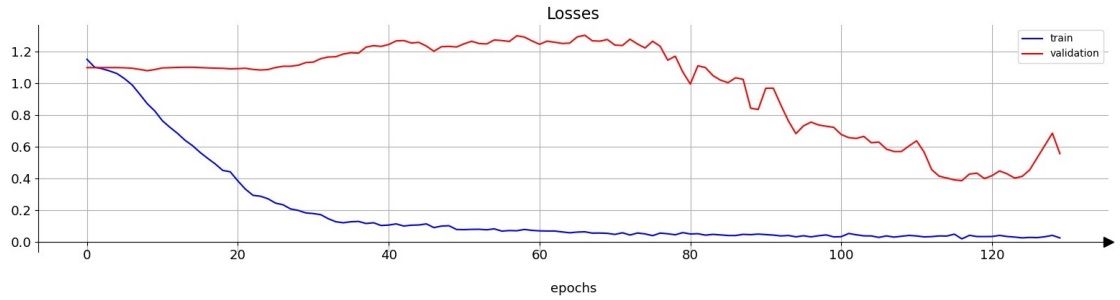


Figure 3.2: Loss for Audio Dataset

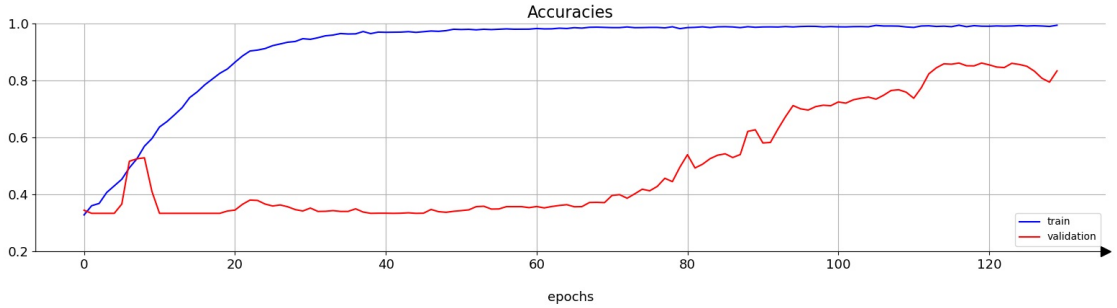
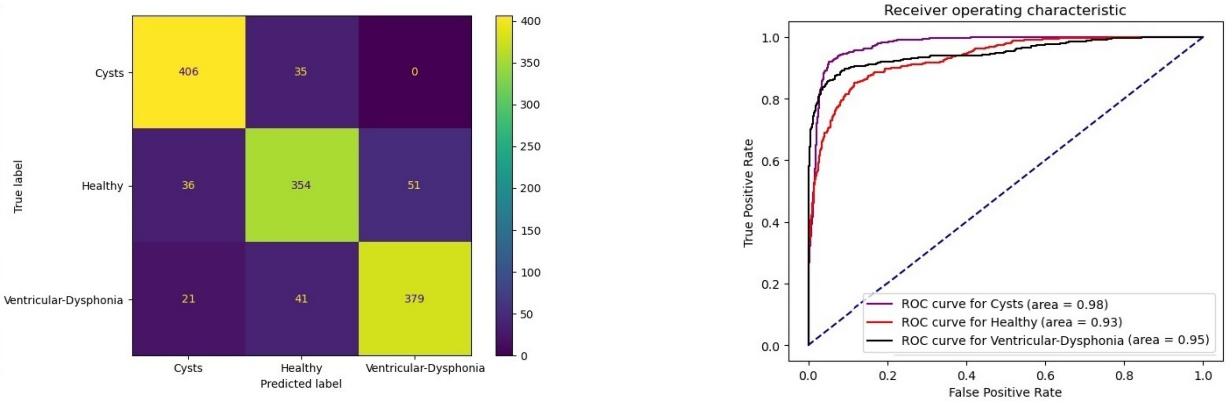


Figure 3.3: Accuracy for Audio Dataset

The best model was saved at approximately the 117^{th} epoch. Figure 3.4 illustrates the results of the audio dataset experiment.



(a) Confusion Matrix for Audio Dataset (b) ROC for Audio Dataset

Figure 3.4: Results of the Audio Dataset

From the confusion matrix, the accuracies per each class can be calculated as follows:

$$class\_acc_i = \frac{CM[i][i]}{\frac{1}{C}\sum_{k=0}^{C-1} CM[k][i]}, \tag{3.1}$$

where C is the number of classes in the confusion matrix and i is the index of the class for which the accuracy is computed. The following accuracies result: Cysts – 92.06%, Healthy – 80.27%, Ventricular Dysphonia – 85.94%.

Electroglottograph Dataset

Figures 3.5 and 3.6 depict the evolution of the training and validation losses and accuracies respectively for 530 epochs on the EGG dataset:

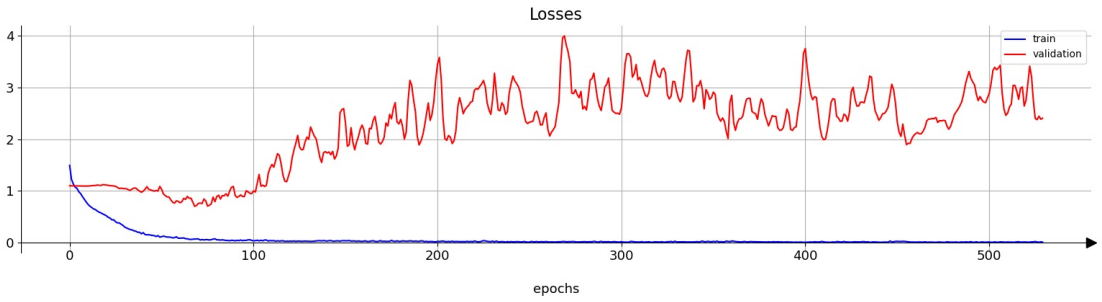


Figure 3.5: Loss for EGG Dataset

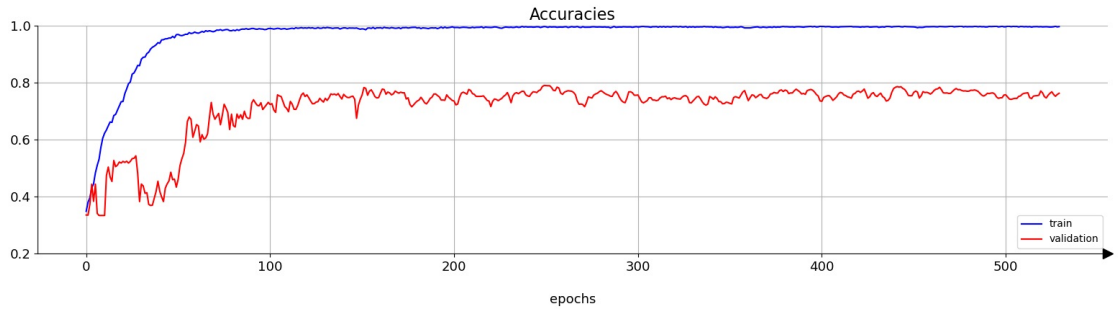
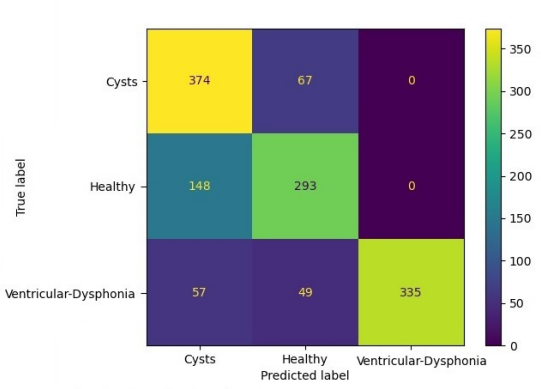
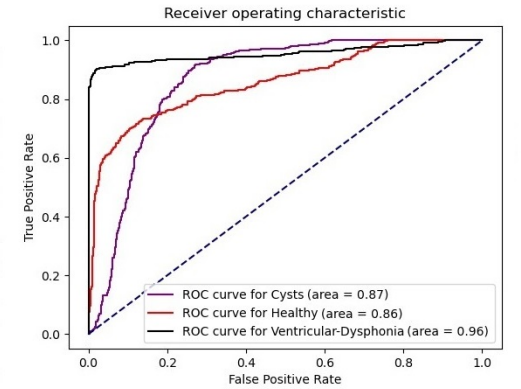


Figure 3.6: Accuracy for EGG Dataset

The best model was saved at approximately the 77^{th} epoch. Figure 3.7 illustrates the results of the EGG dataset experiment.



(a) Confusion Matrix for EGG Dataset



(b) ROC for EGG Dataset

Figure 3.7: Results of the EGG Dataset

From the confusion matrix, the following accuracies result: Cysts – 84.81%, Healthy – 66.44%, Ventricular Dysphonia – 75.96%.

Combined Datasets

The combined dataset pipeline is shown in figure 3.8. The feature tensors were concatenated along the time axis, with the MFCCs extracted EGG signals after the MFCCs extracted from audio. This resulted in tensors of shape (26,16) i.e. matrices with 16 time bins of 26 MFCCs each.

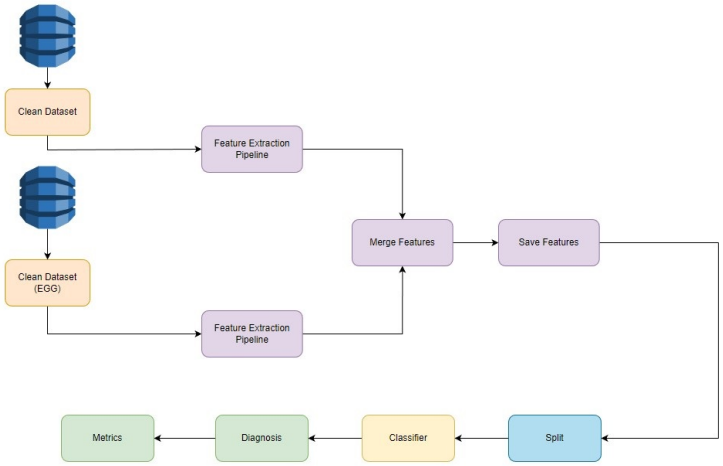


Figure 3.8: Pipeline for Combined Dataset

Figures 3.9 and 3.10 depict the evolution of the training and validation losses and accuracies respectively for 530 epochs on the combined dataset:

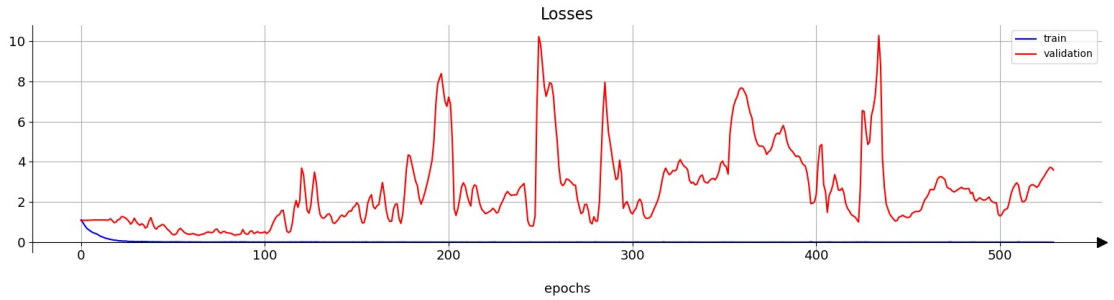


Figure 3.9: Loss for Combined Dataset

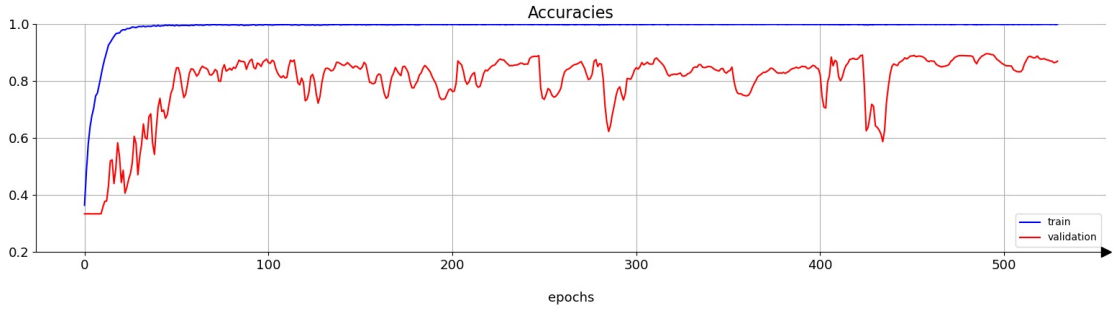
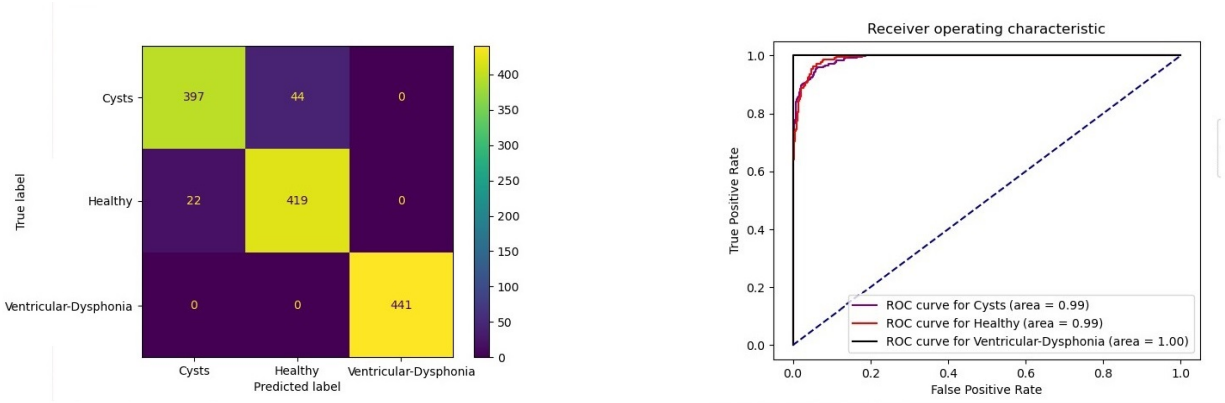


Figure 3.10: Accuracy for Combined Dataset

The best model was saved at approximately the 80^{th} epoch. Figure 3.11 illustrates the results of the combined dataset experiment.

From the confusion matrix, the following accuracies result: Cysts – 90.02%, Healthy – 95.01%, Ventricular Dysphonia – 100%.



(a) Confusion Matrix for Combined Dataset (b) ROC for Combined Dataset

Figure 3.11: Results of the Combined Dataset

Results Comparison and Performance Evaluation

In case of pathology detection, the positive class is the pathological class and the negative class is the healthy class, hence the False Negative is the type of error with the higher cost than the False Negative in the given situation. In the given situation, the False Negative Rate (i.e. FNR) is defined as:

$$FPR = \frac{\sum_{i \neq n} CM[i][n]}{\sum_{i=0}^{C-1} \sum_{j=0}^{C-1} CM[i][j]}, \tag{3.2}$$

where n is the index of the ngative class.

The False Positive Rate (i.e. FPR) is defined as:

$$FPR = \frac{\sum_{i \neq n} CM[n][i]}{\sum_{i=0}^{C-1} \sum_{j=0}^{C-1} CM[i][j]}, \tag{3.3}$$

where n is the index of the ngative class.

Table 3.1 depicts the mean accuracies, FPR and FNR obtained by each of the three experiments:

	Audio	EGG	Combined
Mean Accuracy	86%	75%	95%
FNR	5.74%	8.76%	3.32%
FPR	6.57%	11.18%	1.66%

Table 3.1: Result comparisons for mean accuracies, FPR and FNR

Each experiment exceeded the 71.4% mean accuracy threshold, which is deemed as the mean accuracy of a medical expert by the authors of [14]. The combined dataset approach yielded the best results in terms of mean accuracy and absolute value of FNR. However, the EGG approach yielded the best FNR over FPR ratio, meaning that the likelihood of a false negative error is 1.27 times lower than the likelihood of a false positive, while the combined approach yielded a double likelihood of false negative error compared to the likelihood of false positives.

Chapter 4
Conclusions and Further Development

Personal Contributions

The personal contributions of the student include the modular and flexible design of the proposed algorithm that enables the system for a multi-purpose one-dimensional signal classification (e.g. audio), including but not limited to: music classification, speaker detection or pathology detection. Another personal contribution is represented by the multiple dataset approach that consisted in the concatenation of extracted features from two distinct datasets composed of signals recorded by different means. Furthermore, the flexibility of the algorithm allows the users to perform various experiments with different feature sets or neural network architectures on individual datasets or their combinations. The source code of the project can be found online [1].

Conclusions

The 75% accuracy obtained by the presented system on the dataset obtained by the electroglottographic recordings of human speech, compared to the 86% accuracy of the audio dataset indicates that the use of other features could be explored to attain a better accuracy than the accuracy obtained with MFCC features. Hence, to maximize the accuracy of the system on non-audio signals, a differentiated feature extraction method for each type of signal is worth investigating. Additionally, the fact that the combined dataset approach yielded a higher accuracy than both other methods, separately, indicates that the two types of signals capture complementary information regarding the pathological nature of the recordings.

Furthermore, the implemented algorithm could also prove useful in applications such as non-invasively monitoring vocal recovery after surgical interventions by measuring the dissimilarity between the neural network's current output and a healthy, target output example vector.

The promising results obtained suggest that further research and improvements of the system are legitimate and there are several directions of development that could be explored.

Advantages of the implemented system:

- non-invasive
- ease of use because the patients only have to speak
- objective diagnosis based on vocal parameters

Lastly, given the fact that the proposed algorithm obtained state of the art results by a peak mean accuracy of 95%, the system could prove useful for further research and practical use in the field of

[1] `https://gitlab.upb.ro/robert.bencze/voice-pathology-detection/`

automatic pathology detection by learning more pathologies.

Further Research

For the further development of the system, data augmentation by noise addition could prove useful for applications of remote diagnosis, since home environments and consumer-grade equipments likely differ from the setup employed in the creation of the dataset. Furthermore, because some pathologies of the dataset contain a relatively low number of examples, new artificial examples can be generated by Generative Adversarial Networks [74]. Additionally, to enhance the learning capacity of the network on an augmented dataset, learned features could be extracted from the augmented dataset, using the VGGish network [75] that would have otherwise not been feasible because the VGGish network requires signals that last a minimum of 1s and the dataset used in the experiments required the signals to be windowed due to its fairly reduced size.

Other proposed further development includes the following approaches:

- Removal of the gender bias of some pathologies (e.g. Cysts - biased toward female, Ventricular Dysphonia - biased toward male, see table 2.1) by enhancement with real and artificial examples
- Exploring EMG-audio, EMG-EGG and EMG-EGG-audio combined signal datasets
- Extraction of dataset-specific feature tensors to maximize the per-dataset accuracy of the system
- Performing classification experiments with other neural network architectures

References

[1] Jeff Bilmes, Thomas Richardson, K. Livescu, Peng Xu, Yigal Brandman, Eric Sandness, and Eva Holtz. Discriminatively structured graphical models for speech recognition. -, 06 2022.

[2] Bogdan Woldert-Jokisz. Saarbruecken voice database. -, 2007.

[3] Leyi Wei and Quan Zou. Recent progress in machine learning-based methods for protein fold recognition. *International journal of molecular sciences*, 17(12):2118, 2016.

[4] Amirhossein Kiani, Bora Uyumazturk, Pranav Rajpurkar, Alex Wang, Rebecca Gao, Erik Jones, Yifan Yu, Curtis P Langlotz, Robyn L Ball, Thomas J Montine, et al. Impact of a deep learning assistant on the histopathologic classification of liver cancer. *NPJ digital medicine*, 3(1):1–8, 2020.

[5] Fahad Taha Al-Dhief, Nurul Mu'azzah Abdul Latiff, Nik Noordini Nik Abd. Malik, Naseer Sabri Salim, Marina Mat Baki, Musatafa Abbas Abbood Albadr, and Mazin Abed Mohammed. A survey of voice pathology surveillance systems based on internet of things and machine learning algorithms. *IEEE Access*, 8:64514–64533, 2020.

[6] Soumyadeep Bhattacharjee and Wenyao Xu. Voicelens: a multi-view multi-class disease classification model through daily-life speech data. *Smart Health*, 23:100233, 2022.

[7] Farah Nazlia Che Kassim, Hariharan Muthusamy, Vikneswaran Vijean, Zulkapli Abdullah, and Rokiah Abdullah. Dual-tree complex wavelet packet transform for voice pathology analysis. *Pertanika Journal of Science & Technology*, 28(3), 2020.

[8] Farika Putri, Wahyu Caesarendra, Elta Diah Pamanasari, Mochammad Ariyanto, and Joga D Setiawan. Parkinson disease detection based on voice and emg pattern classification method for indonesian case study. *JEMMME (Journal of Energy, Mechanical, Material, and Manufacturing Engineering)*, 3(2):87–98, 2018.

[9] Meisam Khalil Arjmandi, Mohammd Pooyan, Hojat Mohammadnejad, and Mansour Vali. Voice disorders identification based on different feature reduction methodologies and support vector machine. In *2010 18th Iranian Conference on Electrical Engineering*, pages 45–49. IEEE, 2010.

[10] Fahad Al-Dhief, Nurul Muazzah Abdul Latiff, Marina Mat Baki, Nik Noordini Nik Abd Malik, Naseer Sabri, and Musatafa Albadr. Voice pathology detection using support vector machine based on different number of voice signals. 11 2021.

[11] Fahad Taha Al-Dhief, Marina Mat Baki, Nurul Mu'azzah Abdul Latiff, Nik Noordini Nik Abd Malik, Naseer Sabri Salim, Musatafa Abbas Abbood Albader, Nor Muzlifah Mahyuddin, and Mazin Abed Mohammed. Voice pathology detection and classification by adopting online sequential extreme learning machine. *IEEE Access*, 9:77293–77306, 2021.

[12] Everthon Silva Fonseca, Rodrigo Capobianco Guido, Sylvio Barbon Junior, Henrique Dezani, Rodrigo Rosseto Gati, and Denis César Mosconi Pereira. Acoustic investigation of speech pathologies based on the discriminative paraconsistent machine (dpm). *Biomedical Signal Processing and Control*, 55:101615, 2020.

[13] Sidra Abid Syed, Munaf Rashid, Samreen Hussain, and Hira Zahid. Comparative analysis of cnn and rnn for voice pathology detection. *BioMed Research International*, 2021, 2021.

[14] Jonathan G Richens, Ciarán M Lee, and Saurabh Johri. Improving the accuracy of medical diagnosis with causal machine learning. *Nature communications*, 11(1):1–9, 2020.

[15] Roland Priemer. *Introductory signal processing*, volume 6. World Scientific, 1991.

[16] Camille Jordan. *Cours d'analyse de l'École polytechnique*, volume 1. Gauthier-Villars et fils, 1893.

[17] Jean Baptiste Joseph Baron Fourier. *The analytical theory of heat*. The University Press, 1878.

[18] René Johan Beerends, Henricus G ter Morsche, JC Van den Berg, and EM Van de Vrie. *Fourier and Laplace transforms*. -, 2003.

[19] Pierre Wickramarachi. Effects of windowing on the spectral content of a signal. *Sound and vibration*, 37(1):10–13, 2003.

[20] F.J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[21] Dominic Mazzoni. Audacity® 3.1.3.0, 1999.

[22] Lizhe Tan and Jean Jiang. *Digital signal processing: fundamentals and applications*. Academic Press, 2018.

[23] Mihnea Udrea. Digital signal processing course, 2020.

[24] Vijay K Garg and Yih-Chen Wang. Data communication concepts. *The Electrical Engineering Handbook*, pages 983–988, 2005.

[25] Fredric J Harris. *Multirate signal processing for communication systems*. River Publishers, 2021.

[26] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[27] M Barbero RAI, H Hofmann IRT, and ND Wells BBC. Dct source coding and current implementations for hdtv. *EBU Technical Review*, 1992.

[28] Hee-Joung Kim, Dae-Hong Kim, Chang-Lae Lee, Hyo-Min Cho, Hye-Suk Park, A-Ram Yoo, and Young-Sub Lee. Comparison and evaluation of jpeg and jpeg2000 in medical images for cr (computed radiography). *Journal of the Korean Physical Society*, 56(3):856–862, 2010.

[29] Alejandro Antonio Torres Garcia, Carlos Alberto Reyes Garcia, Luis Villasenor-Pineda, and Omar Mendoza-Montoya. *Biosignal Processing and Classification Using Computational Learning and Intelligence: Principles, Algorithms, and Applications*. Academic Press, 2021.

[30] J Wise. The autocorrelation function and the spectral density function. *Biometrika*, 42(1/2):151–159, 1955.

[31] Gunnar FANT. T. chiba and m. kajiyama, pioneers in speech acoustics(<feature article>;sixtieth anniversary of the publication of the vowel, its nature and structure by chiba and kajiyama). *Journal of the Phonetic Society of Japan*, 5(2):4–5, 2001.

[32] Damien Vincent, Olivier Rosec, and Thierry Chonavel. A new method for speech synthesis and transformation based on an arx-lf source-filter decomposition and hnm modeling. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–525. IEEE, 2007.

[33] David Cohen. Acoustic theory of speech production, with calculations based on x-ray studies of russian articulations, 1962.

[34] Asher Laufer and I.D. Condax. The function of the epiglottis in speech. *Language and Speech*, 24(1):39–62, 1981. PMID: 7266191.

[35] Carol A Fowler and Elliot Saltzman. Coordination and coarticulation in speech production. *Language and speech*, 36(2-3):171–195, 1993.

[36] Vincent L Gracco. Timing factors in the coordination of speech movements. *Journal of Neuroscience*, 8(12):4628–4639, 1988.

[37] Ben Maassen, Raymond Kent, and Hermann Peters. *Speech motor control: In normal and disordered speech*. Oxford University Press, 2007.

[38] Hanspeter Herzel, David Berry, Ingo Titze, and Ina Steinecke. Nonlinear dynamics of the voice: signal analysis and biomechanical modeling. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 5(1):30–34, 1995.

[39] Olaide Agbolade. Vowels and prosody contribution in neural network based voice conversion algorithm with noisy training data. *arXiv preprint arXiv:2003.04640*, 2020.

[40] Erwan Pépiot. Male and female speech: a study of mean f0, f0 range, phonation type and speech rate in parisian french and american english speakers. In *Speech Prosody 7*, pages 305–309, 2014.

[41] Damir Kovacić and Evan Balaban. Voice gender perception by cochlear implantees. *The Journal of the Acoustical Society of America*, 126:762–75, 09 2009.

[42] W Freeman Twaddell. On defining the phoneme. *Language*, 11(1):5–62, 1935.

[43] Fernando Trujillo. English phonetics and phonology, 2010.

[44] Carlos Hernandez-Espinosa, P Gomez-Vilda, Juan I Godino-Llorente, and Santiago Aguilera-Navarro. Diagnosis of vocal and voice disorders by the speech signal. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 4, pages 253–258. IEEE, 2000.

[45] Kathleen M Tibbetts, Laura M Dominguez, and C Blake Simpson. Impact of perioperative voice therapy on outcomes in the surgical management of vocal fold cysts. *Journal of Voice*, 32(3):347–351, 2018.

[46] Jacob Shvero, Rumelia Koren, Tuvia Hadar, Eitan Yaniv, Juditz Sandbank, Raffael Feinmesser, and Rivka Gal. Clinicopathologic study and classification of vocal cord cysts. *Pathology-Research and Practice*, 196(2):95–98, 2000.

[47] Richard Luchsinger and Gottfried E Arnold. Taschenfaltenstimme. In *Handbuch der Stimm-und Sprachheilkunde*, pages 447–451. Springer, 1970.

[48] Celia Stewart, Linda Winfield, Ann Hunt, Susan B Bressman, Stanley Fahn, Andrew Blitzer, and Mitchell F Brin. Speech dysfunction in early parkinson's disease. *Movement disorders: official journal of the Movement Disorder Society*, 10(5):562–565, 1995.

[49] Geralyn M Schulz and Megan K Grant. Effects of speech therapy and pharmacologic and surgical treatments on voice and speech in parkinson's disease: a review of the literature. *Journal of communication disorders*, 33(1):59–88, 2000.

[50] Peter Kitzing. Clinical applications of electroglottography. *Journal of Voice*, 4(3):238–249, 1990.

[51] Kevin Gurney. *An introduction to neural networks*. CRC press, 1997.

[52] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[53] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[54] Will Koehrsen. Overfitting vs. underfitting: A complete example. *Towards Data Science*, 2018.

[55] Jimin Tan, Jianan Yang, Sai Wu, Gang Chen, and Jake Zhao. A critical look at the current train/test split in machine learning. *arXiv preprint arXiv:2106.04525*, 2021.

[56] Yingjie Tian and Yuqi Zhang. A comprehensive survey on regularization strategies in machine learning. *Information Fusion*, 80:146–166, 2022.

[57] Zongben Xu, Xiangyu Chang, Fengmin Xu, and Hai Zhang. $l\_\{1/2\}$ regularization: A thresholding representation theory and a fast solver. *IEEE Transactions on neural networks and learning systems*, 23(7):1013–1027, 2012.

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[59] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[60] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[61] Martín 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, Yangqing 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, 2015. Software available from tensorflow.org.

[62] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25. Citeseer, 2015.

[63] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[64] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[66] Manfred Pützer and Jacques Koreman. A german database of patterns of pathological vocal fold vibration. *Phonus*, 3:143–153, 1997.

[67] Irreducible.

[68] Md Manjurul Ahsan, MA Parvez Mahmud, Pritom Kumar Saha, Kishor Datta Gupta, and Zahed Siddique. Effect of data scaling methods on machine learning algorithms and model performance. *Technologies*, 9(3):52, 2021.

[69] Juan D Rodriguez, Aritz Perez, and Jose A Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):569–575, 2009.

[70] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data augmentation for time series classification using convolutional neural networks. In *ECML/PKDD workshop on advanced analytics and learning on temporal data*, 2016.

[71] KNRK Raju Alluri and Anil Kumar Vuppala. A study on the emotional state of a speaker in voice bio-metrics. In *Advances in Ubiquitous Computing*, pages 223–236. Elsevier, 2020.

[72] Ovishake Sen, Pias Roy, et al. A convolutional neural network based approach to recognize bangla spoken digits from speech signal. In *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, pages 1–4. IEEE, 2021.

[73] Weizheng Shen, Ding Tu, Yanling Yin, and Jun Bao. A new fusion feature based on convolutional neural network for pig cough recognition in field situations. *Information Processing in Agriculture*, 2020.

[74] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

[75] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.