

Voice interference detection

Or Beyar

Dror Arie

**Supervisors:**

Roni Chernyak, Nimrod Peled

**Summary**

1. **Abstract**

Dysphonia is the impairment of voice production as diagnosed by a clinician, often used interchangeably with the complaint of hoarseness, which is a symptom of altered voice quality. While many patients experience dysphonia as a natural part of the aging process, it can be a symptom of a serious underlying condition.

**1.1 Introduction**

The goal of this project continues the work done by previous projects and try and improve the classification results by pre-processing the data in a different way than done previously.  
I will develop an algorithm that determines whether the patient is healthy of suffers from various voice disorders.  
The algorithm will base its decision on features that will be extracted from patients’ audio samples acquired in the Carmel Medical Center. The most informative features will be fed to a classifier that will detect whether the patient suffers from voice disorder and which one it is.

* 1. **Data and relevant Assumptions**

The dataset we used is being collected by medical experts and is presented by an excel sheet containing raw data such as Age, medical history, smoking status etc.

What I will be focusing on is the audio samples taken from the patients.

The samples contain:

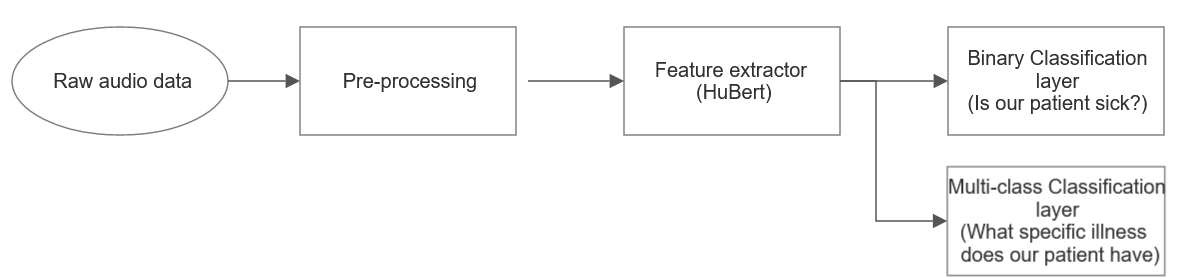
* 3 different vowel sounds (Ah, Eeh, Ooh).
* 3 pitch frequencies (High, low, and normal voice).
* A reading assignment roughly 30 seconds long.

An important thing to note is that our dataset is unbalanced, There are more than twice more sick patients than healthy ones.   
Hence requiring some balancing techniques such as over and under sampling.

A graph with blue rectangular bars

Description automatically generated with medium confidence

*Figure 1 - Labels distribution depicting the unbalanced dataset*

1. **Project pipeline’s Block diagram**

2.1 **Modules**

**Cleaning irrelevant data – Some of the recordings are mis-labeled or not labeled at all. Mis-labels are easy to fix (mostly typos) but data that is not labeled at all was discarded.**

**Mono channel and resampling** – Most models require the audio to be mono-channel and sampled at 16 [kHz]. We’ll need to down-sample our audio to be able to work with it.   
**Voice/speech activity detection model** - A big part of the pre-processing is isolating the parts containing actual speech. Our samples consist of patients making various consecutive vowel sounds. In between these sounds there is random noise that we would like to filter out. To do so, we’ll use a VAD/SAD approach to detect speech parts and isolate them.

**Up/down sampling –** Before I feed the data into a model to fine-tune it, I will balance it using up/down sampling technique.

**Feature extractor** – A pre-trained model that will extract features from the audio samples.

**Classification** **layer** – a model (perhaps a small one) that performs binary classification at first, and multi-class in the future.

1. **Pre-Processing**

**3.1 mono-channel and resampling**

This is the easy part; The data consists of multiple .wav files.

I used the program [SoX](https://www.researchgate.net/publication/291119036_SoX_Sound_eXchange) in order to mono-channel and re-sample the files into 16 [kHz].

* 1. **VAD model**

This took some research, but eventually [Google’s webRTC VAD](https://github.com/wiseman/py-webrtcvad) was the chosen tool for this assignment.

This model is not classical, it uses a pre-trained GMM to calculate the probability that a given audio chunk is speech or not speech.

I use it as a black box VAD, but it does have a few hyperparameters that I trial-and-errored with:

**Frame duration-** The frame duration in milliseconds.

**Padding duration-** The amount to pad the window, in milliseconds.

**Aggressive –** The “accuracy” of the model, it is a black-box sort of hyper-param, but after running various times with different values and going into the source code, I concluded that it trades-off precision for speed.

**Minimal chunk length** – The lower bound for how short a vowel can be. Set manually by me based on observations.

* 1. **Up-sampling**

As mentioned before, the data is unstable, meaning the number of sick patients outnumber healthy patients almost 2:1. To address this problem, the data’s minority class, healthy patients, will be up sampled to equalize both labels.

A comparison of a bar graph

Description automatically generated

1. Feature extractor and classification

In this section, I researched with different pre-trained models that are open-source and are easy to fine tune and are ideal for our objective.

4.1 YamNet

YAMNet is a deep net that predicts 521 audio event classes from the [AudioSet-YouTube corpus](http://g.co/audioset) it was trained on. It employs the [Mobilenet\_v1](https://arxiv.org/pdf/1704.04861.pdf) depthwise-separable convolution architecture.

It was trained with audio features computed as follows:

* All audio is resampled to 16 kHz mono.
* A spectrogram is computed using magnitudes of the Short-Time Fourier Transform with a window size of 25 ms, a window hop of 10 ms, and a periodic Hann window.
* A mel spectrogram is computed by mapping the spectrogram to 64 mel bins covering the range 125-7500 Hz.
* A stabilized log mel spectrogram is computed by applying log(mel-spectrum + 0.001) where the offset is used to avoid taking a logarithm of zero.
* These features are then framed into 50%-overlapping examples of 0.96 seconds, where each example covers 64 mel bands and 96 frames of 10 ms each.

These 96x64 patches are then fed into the Mobilenet\_v1 model. The outputs are averaged to give a 1024-dimension embedding, then put through a single logistic layer to get the 521 per-class output scores corresponding to the 960 ms input waveform segment.

* + 1. YamNet Results

Training the classifier head with different optimizers and learning rates and plotting the train set accuracy/loss (TODO change to validation accu/loss when results actually mean something)

A comparison of different colored lines

Description automatically generated

Although the difference is not big, Adam with a learning rate of 0.001 or 0.0005 is ideal.

I re-trained the classifier with a learning rate of 0.001:

A graph of a model

Description automatically generated with medium confidence

And the accuracy on the test set is:

Loss: 2.618

Accuracy: 0.607

# סיכום

We automatically separated the patients’ recordings to achieve better accuracy when we built the identification algorithm.

We used Google’s VAD; we focused mainly on splitting the phonemes and left the sentence completely intact at the end.

To test our VAD’s error rate (falsely labeling noise as a phoneme or missing a phoneme and labeling it as noise) we compared the outputted audio chunk’s MFCC features using DTW:

In the end, after many attempts, parameters, and different methods, the scores generated by comparing these features weren’t consistent enough and we could not determine if the VAD works properly for all recordings. We resorted to manually checking a sub-set of outputs and determined the VAD does isolate the phenomes properly roughly 78% of the time.

15 recordings were properly split out of the 19 we manually checked.

# רשימת מקורות

|  |  |
| --- | --- |
| [1] | B. Gustavii, How to Write and Illustrate a Scientific Paper, 2nd ed., Cambridge, UK: Cambridge University Press, 2008. |
| [2] | E. Bullkich, I. Ilan, Y. Moshe, Y. Hel-Or and H. Hel-Or, "Moving Shadow Detection by Nonlinear Tone-Mapping," in *Proc. of the 19th International Conference on Systems, Signals and Image Processing (IWSSIP 2012)*, Vienna, 2012. |
| [3] | Y. Moshe and H. Hel-Or, "Video Block Motion Estimation Based on Gray-Code Kernels," *IEEE Trans. on Image Processing,* vol. 18 , no. 10, pp. 2243-2254, 2009. |
| [4] | "Signal and Image Processing Lab.," 2018. [Online]. Available: http://sipl.technion.ac.il/. |

תיעוד דרכים שניסינו להתמודד עם הבעיה הזו ואתגרים במהלך הדרך

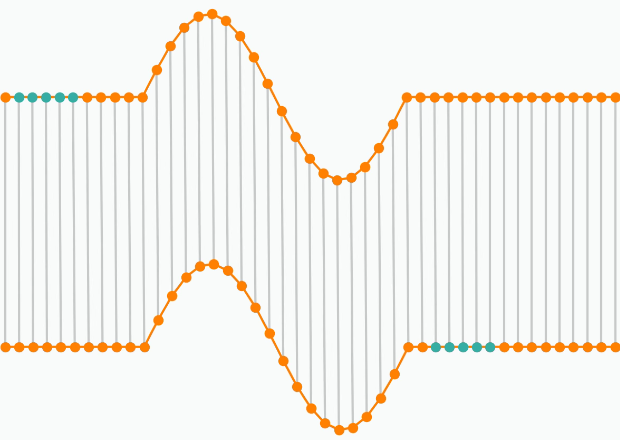
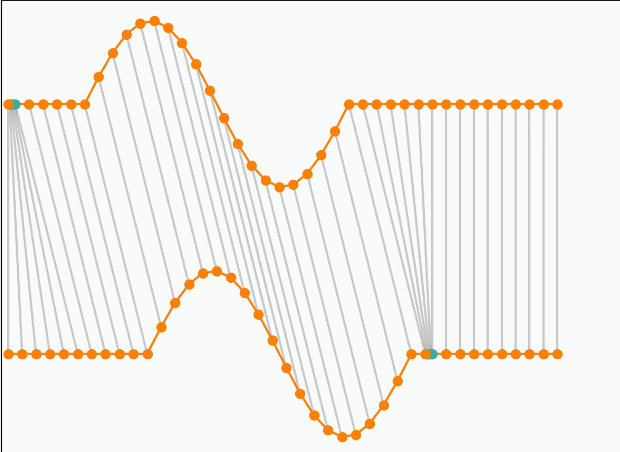
* **PyDub VAD**: PyDub is a Python library for audio processing, and it provides a simple VAD implementation.
* **WebRTC VAD**: WebRTC VAD is an open-source VAD library developed by Google. You can use the **webrtcvad** Python library to integrate it into your projects.
* **Spleeter**: Spleeter is a pre-trained deep learning model for audio source separation, which can be used for VAD as well. It's part of the Deezer Spleeter project.
* **Google Cloud Speech-to-Text**: Google's Speech-to-Text API can be used for VAD by analyzing the transcriptions it provides. You'll need a Google Cloud account and API key to use this.
* **Librosa + Custom Thresholding**: You can use the Librosa library for audio analysis and apply custom thresholding for VAD based on the audio's amplitude.
* **Kaldi**: Kaldi is a popular open-source toolkit for speech recognition, but it also includes VAD functionality. It provides a comprehensive set of tools for speech processing.
* Naïve power -
* Google VAD -
* Speech recognition -
* Mfcc - **Mel-Frequency Cepstrum Coefficients –** A method to represent audio signals as a small set of features that describe the overall shape of the spectral envelope of the signal.  
  It extracts the log of power spectrum of the FFT,  
  project to Mel scale (which is a human-perception scale of pitches)  
  and then a DCT is performed on the output.

A diagram of a graph

Description automatically generated

* DTW - **Dynamic Time Warping -** an algorithm for measuring similarity between two temporal sequences, which may vary in speed and size – this metric is invariant to temporal shifts.  
  Dynamic Time Warping is equivalent to minimizing Euclidean distance between aligned time series under all admissible temporal alignments.

In the figure - Cyan dots correspond to repetitions of time series elements induced by the optimal temporal alignment retrieved by DTW.



Higher Euclidean distance

Lower Euclidean distance