MACHINE LEARNING FOR AUTOMATED LECTURE AUDIO PROCESSING

by

Benjamin Rotker

Commonwealth Honors College

Department of Electrical and Computer Engineering

UNIVERSITY OF MASSACHUSETTS, AMHERST

2023

Table of Contents

I. Introduction

i. General Overview

Misophonia is a mental condition that activates a negative emotional response to specific auditory stimuli [1]. Examples of these stimuli include pen clicking, throat clearing, sniffling, or typing on a keyboard, which ignites feelings of anger and anxiety in individuals with misophonia. These stimuli are more commonly referred to as “trigger sounds” because they heavily interfere with one’s ability to perform seemingly normal daily tasks, such as eating in a busy dining hall, or concentrating on the speaker in a crowded lecture hall. Despite how these symptoms tremendously affect a misophonic’s daily life, this paper is probably the first time the reader has ever heard of this condition. As J. Brout et al. describe in their 2018 paper on misophonia, “Although syndromal features have begun to be characterized empirically, misophonia has not been formally recognized as a specific type of neurological, audiological, or psychiatric disorder” [1]. Considering how recently research has begun on this condition, the lack of official recognition is not surprising.

Based on this observation, one would expect efforts within this thesis to investigate the underlying psychological aspects of misophonia, and to bring more attention to the condition. This project, however, will focus very minimally on these aspects. On the contrary, this thesis seeks to develop software incorporating machine learning to help eliminate these sounds from university lecture recordings. In doing so, it hopes to make learning a more pleasant and efficient experience for those who are distracted by trigger sounds. Additionally, it will argue that the automatic removal of sounds like coughs, throat clears, and lengthy pauses benefits everybody who views lecture recordings, even if they do not have adverse emotional reactions to, or even notice these noises.

ii. Scientific question

This thesis seeks to answer two scientific questions, which are dictated by the following design goal: to create a machine learning model to differentiate between wanted and unwanted sounds in lecture recordings, and to flag unwanted sound segments within the lecture recording audio files. Several solutions exist that automatically trim video audio in portions where no speech is detected, with the purpose of making the video more concise and providing ease of editing to the creator [2] [3]. None of these solutions, however, are tailored specifically to misophonia. Subsequently, the first question that stems from the design goal is: *how can we reshape automatic audio trimming for people with misophonia, while having the result benefit everybody's lecture watching experience, regardless of their aversion to these trimmed out sounds?*

The second question arises from the need to test this machine learning model for accuracy. *How can we evaluate a machine learning model in its ability to successfully trim unwanted audio signals from a lecture recording?* A number of metrics will be explored and calculated for the model trained in this thesis in order to assess its efficacy. Additionally, as discussed in Kawahara et. al, who studied feature selection for lecture audio processing, a machine learning model may perform well with one professor and horribly with another, given how widely speech patterns vary between individuals [14]. Likewise, attention will be given to the design tradeoff between focusing a machine learning model on one professor's speech patterns for better clarity, versus the reduced amount of data resulting from testing on only one professor's lectures per model. This will provide more insight as to why the model achieves the performance metrics it does, which are discussed in the results section.

iii. Significance

The final deliverable of this thesis will be a ML model trained using supervised learning to detect undesired sounds like coughs, throat clears, and long pauses. The performance results and comparison with gold standard data will be submitted alongside the model itself. Additionally, a lecture video before and after applying the model will be shown to demonstrate the latter's increased conciseness and decreased length. This concept is challenging to envision given an on-paper description; the two videos below demonstrate the idea in a clearer fashion.

The first video is an unedited lecture on the Discrete Fourier Transform from Professor Steve Brunton at the University of Washington. Paying attention to Professor Brunton's speech patterns, he can be observed smacking his lips, clearing his throat, taking breaths, and even having a drink of water in the first minute [4]. Although the typical viewer will not be affected by, or even notice these sounds, the sounds make watching the video a challenge for those with misophonia. The second video is a more abstract view of the Fourier transform from Grant Sanderson of the 3blue1brown youtube channel. Comparing the audio of these two videos, it is obvious that Grant trimmed out breaths between sentences carrying useful information, resulting in a more succinct, smooth flowing narration to listen to [5].

The goal of this comparison is not to shame Professor Brunton in any way for leaving his video unedited, as all these noises are completely natural, and are typical of any person's speech patterns. Conversely, it is to demonstrate how easy following the concepts of a lecture is when the unneeded segments of audio are trimmed out. Additionally, one would not need to watch either of these videos to understand that an algorithm that automatically trims these noises out is much more desirable than doing the process manually, through cutting out each segment of audio by hand with an audio-visual editing software.

The above explanation only highlights the significance of the design goal. The first scientific question, which stems from this goal, asks how we can edit lecture videos to give people with misophonia a better learning experience without cutting out any useful information from the video. This question's significance is governed by the segmentation and classification scheme employed in this project. In order to satisfy this question, all semantic sounds within a lecture, meaning spoken sounds that carry meaning, must be classified as wanted sounds that the model is trained to keep in the recording. Meanwhile, the model should apply an "unwanted sound" class label to all noises that carry no meaning, and can be removed from the audio without smashing two adjacent sentences together too aggressively. This classification scheme is more thoroughly described in *section III.*

The significance of the second scientific question, regarding how to assess our model, is based on the need to depict quantitatively how the model performed. Without the metrics that are typically used in research to evaluate ML models, there would be no concise indication of how the model did. Calculating these metrics accomplishes two primary goals: first, it collects and visualizes data on the performance of a specific type of ML model for audio discrimination, which is a popular area of research in the academic literature on ML and signal processing. Second, the ML model is presented with a lecture audio file, and must make decisions on whether or not to keep segments of that audio based on the chosen features; the resulting accuracy of these decisions depicts how well the chosen model and features make them under uncertainty. Additionally, both ML and its underlying probability theory are core concepts that were explored in the first semester of the year-long seminar taught by Professor Pishro-Nik.

II. Summary of Work of Previous Researchers

i. Introduction

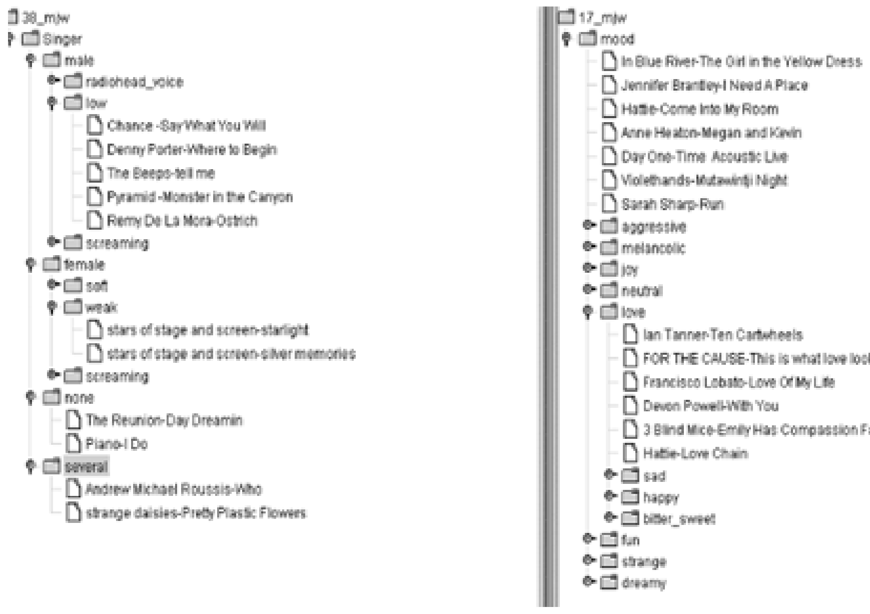
The dream of storing seemingly infinite information on computers has become a reality for the hardware that holds it; consequently, it has become a nightmare for the software that retrieves, sorts, and analyzes it. The unbelievable magnitude of data flowing through modern computers demands efficient yet thorough algorithms to process it correctly. The Spotify music streaming app, for example, processes 600 gigabytes of user generated data every day to enhance customer experience, which equates to 4.8 trillion bits, or ones and zeroes flowing through Spotify's database daily [8]. Luckily, we can scan and evaluate this tremendous amount of data efficiently by using certain mathematical tools in an intelligent way.

Given a basic understanding of audio signal properties, a well-trained individual could manually process a signal set with only a laptop, but that would prove useless due to the enormous amount of audio data available. Conversely, computers can adequately do the math and decision making necessary for evaluating a large number of signals on their own, provided that they are trained properly. This is the foundation of machine learning (ML) for audio signals. The retrieval and analysis of audio signals from a database or dataset is hereafter called audio classification. A successful ML algorithm can make its own decisions in classifying audio, an attribute that grows in efficiency if the model is fed better features based on the data. The various challenges in implementing a secure ML algorithm will be discussed before furnishing the protocol for picking a good dataset and features. Finally, after these core concepts of ML have been summarized, a recent project using ML for lecture video editing will be explored.

ii. Challenges with Machine Learning for Audio Data

In his 2016 paper on audio classification for machine learning, Rong claims the most prevalent issue in any kind of media information retrieval is locating the desired information rapidly and accurately. By extension, achieving this for audio files requires clever attention to detail due to their unstructured organization, complex structure, enormous data, and lack of semantic description [9]. Homburg et al. paint a clearer picture of this complicated structure. While picture data depends on a simple, fixed number of pixels, the size of audio data is determined by how many values are plucked out of the sound wave. This examination of a discrete amount of evenly spaced points in a dataset is more commonly known as sampling. If a signal is sampled at a modest 44,100 times per second, or Hz, a three-minute audio recording is roughly 8 million sampling values in length [10]. Adequately processing a file of this size requires efficient ML algorithms, the building blocks of which will be described in the next three sections.

Homburg et al. also highlight a new branch of issues associated with audio classification based on user-oriented applications. If a ML algorithm wishes to derive song recommendations based on a user's current playlists, different features will be more or less accurate relative to how the user classifies those playlists [10].



- 38_mjw
  - Singer
    - male
      - radiohead_voice
      - low
        - Chance -Say What You Will
        - Denny Porter-Where to Begin
        - The Beeps-tell me
        - Pyramid -Monster in the Canyon
        - Remy De La Mora-Ostrich
      - screaming
    - female
      - soft
      - weak
        - stars of stage and screen-starlight
        - stars of stage and screen-silver memories
      - screaming
    - none
      - The Reunion-Day Dreamin
      - Piano-I Do
    - several
      - Andrew Michael Roussis-Who
      - strange daisies-Pretty Plastic Flowers

- 17_mjw
  - mood
    - In Blue River-The Girl in the Yellow Dress
    - Jennifer Brantley-I Need A Place
    - Hattie-Come Into My Room
    - Anne Heaton-Megan and Kevin
    - Day One-Time Acoustic Live
    - Violethands-Mutawintji Night
    - Sarah Sharp-Run
    - aggressive
    - melancolic
    - joy
    - neutral
    - love
      - Ian Tanner-Ten Cartwheels
      - FOR THE CAUSE-This is what love looks like
      - Francisco Lobato-Love Of My Life
      - Devon Powell-With You
      - 3 Blind Mice-Emily Has Compassion Fatigue
      - Hattie-Love Chain
      - sad
      - happy
      - bitter_sweet
    - fun
    - strange
    - dreamy

Fig. 1. Two examples of user classification schemes. Source: [10]

As seen in figure 1, classification schemes based on singer gender (left) and song mood (right) produce different hierarchical representations of the same songs [10]. There exists a massive range of possible classification styles beyond these two, suggesting that the ML algorithms should be robust enough to satisfy any case [10]. Granted, this phenomenon will become clearer in the *Features for ML Models* section, but for now it suffices to show how user preferences affect ML algorithm complexity.

Guo et al.'s research on support vector machines (SVM) not only confirms the claims made by [9] and [10], but they also propose two new issues arising from audio retrieval and classification. While they agree with Homburg et al. that retrieving data from different distribution patterns in audio classes leads to different results, they also identify two new constraints. Namely, audio retrieval performance is greater for a less varied distribution of audio samples than a scattered one, and the overall accuracy of retrieval is poor [11]. They attribute all three of these problems to the use of conventional Euclidean distance between audio types, or classes, for measuring similarity in audio patterns [11]. This phenomenon is better visualized with the following figure:

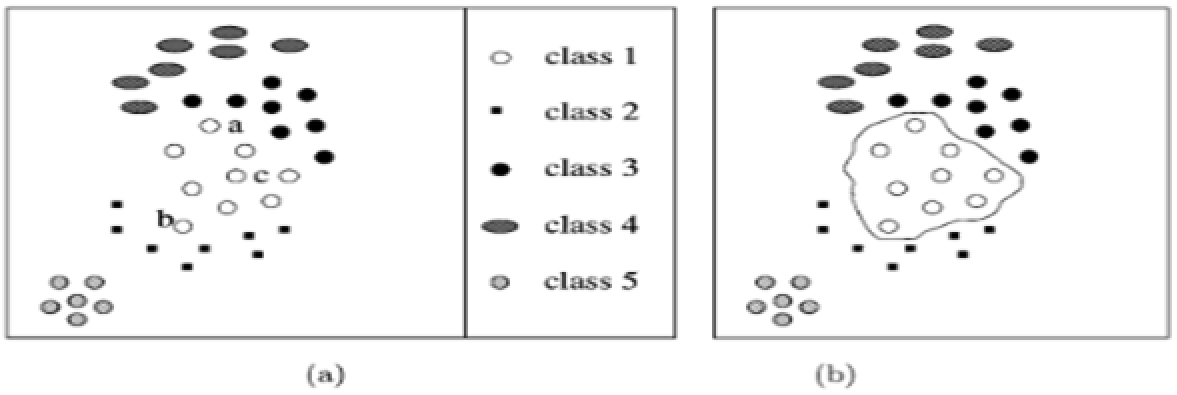


Fig. 2. Varied audio signal distribution patterns. Source: [11].

To reiterate, when classes of audio signals are distributed within a 2D plane, the classification becomes more accurate when similar audio signals are clustered [11]. In more homogeneous distributions, however, performance suffers, and the ML algorithm needs to account for this to be successful.

Kawahara et al. introduce another aspect that contributes to the intricacies of audio files depicted in the previous three papers. Although automatic speech recognition algorithms accurately transcribe audio from grammatically correct readings such as audiobooks with ease, spontaneous speech from scenarios like conversation and public speaking gives these algorithms trouble [14]. The collection of spontaneous Japanese (CSJ) used in this project derives from technical conference presentations and monologue talks. 224 total presentations are used with only experienced male Japanese speakers who did not use drafts. As Homburg et al. clarify in

[10], audio classification algorithms perform differently when applied to different audio. Hence, the developed model may not work well with less experienced speakers, female speakers, or speakers who used a draft.

In spontaneous lecture speech, the speakers naturally make the data difficult to deal with. They alternate randomly between long bouts of uninterrupted, informational speech and speech with frequent, meaningless pauses [14]. The lengthier inputs caused by these patterns are challenging for speech decoding. As a solution, the researchers revise their speech processing algorithm to simultaneously recognize and subdivide speech based on these pauses, or lack thereof.

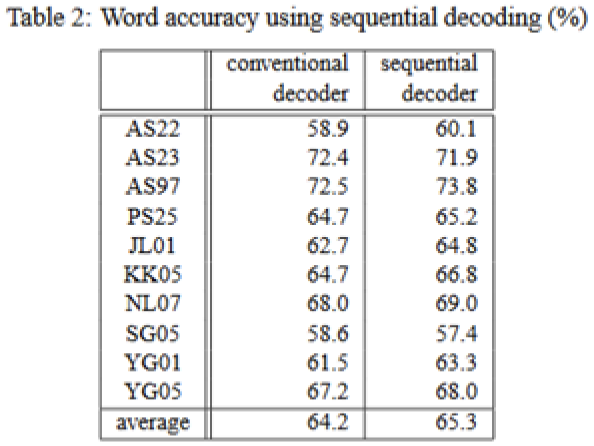


Table 2: Word accuracy using sequential decoding (%)

	conventional decoder	sequential decoder
AS22	58.9	60.1
AS23	72.4	71.9
AS97	72.5	73.8
PS25	64.7	65.2
JL01	62.7	64.8
KK05	64.7	66.8
NL07	68.0	69.0
SG05	58.6	57.4
YG01	61.5	63.3
YG05	67.2	68.0
average	64.2	65.3

Fig. 3. Word identification accuracy improvements with sequential decoding. Source: [14]

The figure above depicts the accuracy improvements of this decoding algorithm, called sequential decoding, over the previous algorithm, which cuts out these pauses before processing the words [14].

Despite underlining several challenges of audio classification with ML, this section pinpoints only one solution, which describes Kawahara et al.'s work with sequential decoding. The issues explained by [9], [10], and [11] focus primarily on feature selection and optimization. Correspondingly, their solutions will be described more succinctly in *Features for Machine*

Learning Models and *Optimal Feature Selection.* Before this, however, the datasets used for feature selection must be discussed.

iii. Datasets for Machine Learning Models

Running a ML algorithm without a well-defined dataset is analogous to driving a car without an engine or gas. Datasets are collections of data subdivided into training data and testing data. The training data, which teaches the ML algorithm to make its own decisions based on the algorithm's purpose, is represented by the engine. Likewise, the testing data is fed to the ML model to run it and test its performance, like fuel in a car. As was demonstrated in the 3rd module in 499CM, a ML model cannot be tested with the same data it was trained with, as the performance results would be deceptively good. Furthermore, a poor or nonexistent dataset demolishes the ML model's functionality.

Every paper in this review except for Sharma et al.'s meta-analysis tests a ML algorithm on a dataset [15]. Intuitively, each of the remaining seven papers uses a different dataset, as they all test slightly different aspects of ML for audio signal classification. These datasets are often borrowed from public audio file databases. In [9], Rong uses two audio databases to evaluate ML model performance with statistical learning algorithms called support vector machines (SVMs): the General Sounds database, comprised of isolated sounds like applause and car sounds, and the Audio Scenes database, containing combined sounds like parks and restaurants [9] [11]. Guo et al. also test SVMs, and they do so with a different database called Muscle Fish that also contains isolated combined sounds [11]. Both studies observe the effects of SVMs on audio classification algorithms; they vary only in their implementation and discussions [9] [11].

Conversely, Homburg et al. test their model solely on song audio clips. This is because they wish to study metadata, or supplementary data that describes given data. In Homburg's case, metadata like song title, genre, and lyrics complement the raw song files [10]. The design of ML models to handle data and metadata adds complexity and is formally called "Multi View Learning" [10]. The presence of this specific metadata allows for the multitude of classification opportunities outlined in figure 1 and expanded upon in *Challenges with Machine Learning for Audio Data.* Although this broad of a dataset requires a bulletproof ML model for success in audio classification, the fact that it can create such a robust model is impressive and accentuates how a more well-defined dataset is needed to handle more complex cases [10].

Published in 2000, Beth Logan's paper on music modeling is the oldest of all the papers in this review. It follows that the scope of her research was narrower than Homburg et al.'s classification of music metadata, or Rong and Guo et al.'s separation of isolated and mixed sounds [9] [10] [11]. Logan trains a ML model to simply differentiate between music and audio signals within the same audio file. She constructs her dataset with a 3-hour audio segment of a broadcasted news show containing interviews, commercials, and music [12]. The data is further portioned into 120 minutes of training data and 40 minutes of testing data [12]. Kim et al. also tests a ML model's ability to discriminate speech and music data: they incorporate audiobook files from Korean, English and Chinese CDs for read speech data. In addition, they include a broadcast dataset involving music, interviews, and laughter, much like Logan's dataset [12] [13].

Although the different dataset choices and their underlying motivations are now clear, there remains one intermediate topic before their respective performance results within ML models can be examined. The features extracted from a dataset, which can be thought of as its most defining characteristics, must be thoroughly explained [15]. Since these models make decisions based on dataset features rather than the complete raw data, the motivations behind choosing features must be observed to understand their gains for model efficiency. The last section bridges this gap; the characteristics of features and their implementation in the reviewed studies will be explained before concluding with their performance gains.

iv. Feature Selection for Machine Learning Models

The more training a model is provided, the more intelligent it will be. The features used to train a signal set are preferred because they are significant enough to distinguish different signals yet concise enough to not overwhelm the algorithm with too much data [15]. Sharma et al. define features more strictly as “the compact representation of a signal”. One example of features extracted from the time domain is shown below:

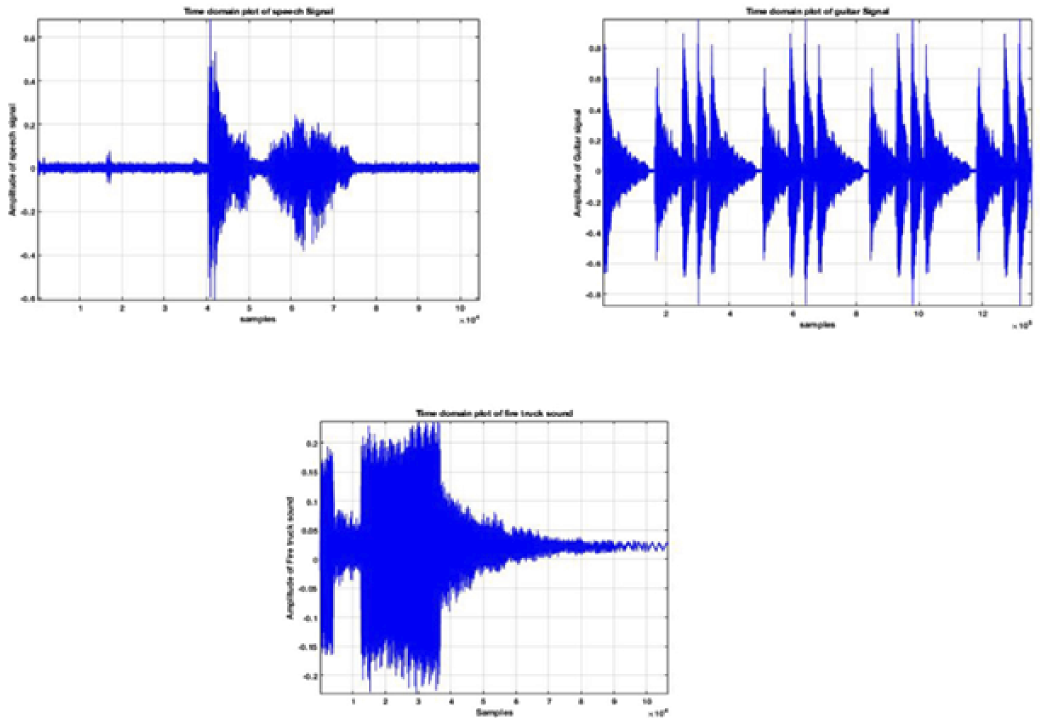


Fig. 4: Time domain plot of speech, guitar, and fire truck sounds. Source: [15]

Figure 4 shows the amplitude of three signals over time. The amplitude, or loudness, has distinct patterns for each of these three sounds. As seen in the top left box, speech has a continuous, smooth envelope with varying amplitude, while the guitar strum in the top right has a consistent patterned structure with frequent spikes. The fire truck sound at the bottom has a brief duration and high amplitude [15].

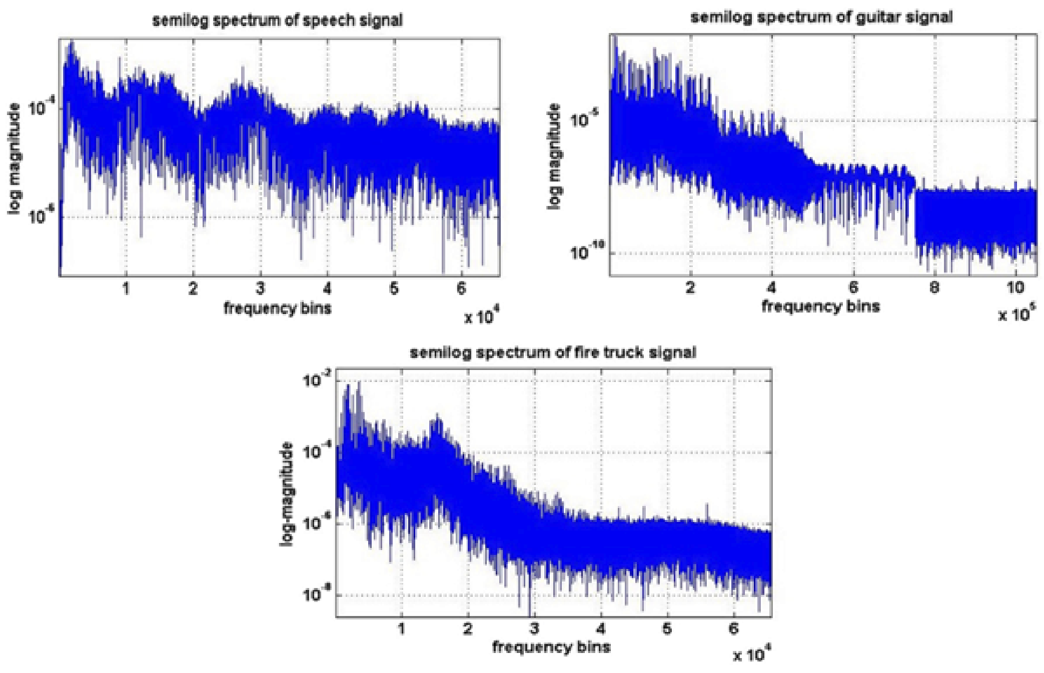


Fig. 5: Frequency domain plot of speech, guitar, and fire truck sounds. Source: [15]

Figure 5 gives the same three signals plotted in the frequency domain. Here, the magnitude range of the speech frequency spectrum is much less than that of the firetruck (environmental) and guitar frequency spectra [15]. Characterizing these two spectra serves as the basis for feature selection. If we can identify patterns that help differentiate a signal based on its characteristics, we have found a potentially successful feature for ML that costs less than analyzing an entire sound file.

The three most prominent audio features studied in the other seven papers besides Sharma et al. are as follows: The zero-crossing rate (ZCR) of an audio signal is the number of times that signal switches from positive value to negative value or vice versa in each time period [15]. ZCR features are useful, as they identify whether a signal frame contains speech, no speech, or pure silence. The ZCR tends to be higher for unvoiced portions of an audio signal, making it an excellent discriminator for speech recognition [15]. Short time energy (STE) is the average energy per audio frame; it is naturally higher in voiced frames than in unvoiced frames, making it stand out as another speech discrimination algorithm [15]. Finally, Mel-frequency cepstral coefficients MFCC, which describe an audio clip's short-time power spectrum, are the most common features used out of the three [15]. MFCCs imitate the human auditory system closely due to the spacing of its frequency bands, so they work well in audio classification ML algorithms that seek to imitate human hearing and decision making [15].

Sharma et al. describe the MFCC extraction algorithm as follows: frame a given audio signal into short frames, incorporating windowing. Windowing isolates mostly stationary portions of a signal in time from the complete non-stationary audio signal [15]. The simplest form of windowing is rectangular windowing, in which a rectangular box is slid over one of these quasi-stationary signal segments; only the signal portions inside the rectangle are analyzed at a time. The rectangular box is slid across the entire signal until it is completely analyzed, one box at a time.

Next, the periodogram estimates of the power spectrum are calculated for each frame. The Mel-filterbank is then applied to each power spectrum estimate before summing each filter's energy. Finally, the discrete cosine transform is applied on the logarithm of each filterbank energy. This provides the desired MFCC coefficients, a subset of which is selected as features for the machine learning algorithm [15]. This exact process is depicted and used in B. Logan's research on MFCCs [12].

Most studies in this review select time and frequency-based features for their machine learning models. F. Rong utilizes time-based features ZCR and STR, as well as frequency based MFCCs. MFCCs are also thoroughly studied in the following papers: Homburg et al. apply MFCCs to music classification [10]. Kawahara and his colleagues develop a more efficient lecture speech processing method upon applying MFCC, differential MFCC, and differential power features to their ML algorithm [14]. Furthermore, in 2007, Kim et al. implemented Mel-cepstrum modulation energy (MCME) features and observed an error reduction rate of 71% over 4 Hz modulation energy (ME) features and 74% over 8 Hz ME features [13]. Kim and their colleagues accelerate off findings from Logan's 2000 paper, which proclaims MFCCs as a fundamental feature for audio signal processing [12]. Sharma et al. identify time and frequency-based features as a prominent mathematical tool in audio signal processing over the last 70 years. Despite the prevalence of these features, Sharma et al. highlight deep learning features as a prominent tool that has only recently been brought to light [15].

The only paper that magnifies deep features through a practical lens is Wichern et al.'s 2018 study on single-channel speech separation. Not only do these two researchers from Mitsubishi's electric research lab improve upon other papers by examining the iterative reconstruction of audio signal phase, but they implement this reconstruction as a neural network layer [16]. Neural networks are webs of data that mimic the human brain in structure. They supplement machine learning algorithms to aid in more efficient problem solving [8]. Neural networks are a subset of deep learning models. Sharma et al. describe neural networks in their meta-analysis, emphasizing how hidden features are discovered within deep learning models.

With the correct tools, powerful features for machine learning can be extracted from low level data [15].

v. Similar Solution

The aforementioned papers cover several vital ML concepts, making their review an excellent precursor to any audio processing project that incorporates ML. Despite this, none of them align with the research direction of this thesis project quite like Devandran Govender's 2018 master's thesis, *Investigating audio classification to automate the trimming of recorded lectures.* Using audio signal classification (ASC), Govender's project explores whether a ML model can subdivide a lecture recording audio signal into classes that represent either lecturer speech or student chatter. This classification helps universities trim the beginnings and ends of lecture recordings out, so the edited recordings contain only the useful portion where the lecturer is the sole speaker [17]. Govender's methodology is similar to that outlined in F. Rong's paper, where audio data features are extracted before classifying the audio data with a SVM [9].

Govender chooses two classification classes for his study: speech, represented by a dominant voice like the lecturer presenting to students, and non-speech, like multiple students talking [17]. Conversely, this thesis classifies between lecturer speech, which contains useful information, and lecturer non-speech, like lip smacks or stutters. While Govender's study is concerned with noises made by both the speaker and anyone else in the classroom, this thesis focuses on sounds made solely by the lecturer. Likewise, Govender's ML model automatically flags the beginning and end of lecturer speech, while this thesis' model strives to pinpoint the smaller, in-between segments of lecturer non-speech that can be removed [17].

Regardless of these minor differences, Govender describes a detailed methodology that is both inspired by the principles outlined in this review, and highly applicable to this overall project. He uses the pyAudioAnalysis Python library because it is free, and offers flexible functionality for ML. Most notably, this library can extract audio features, train classification models, derive performance metrics, and segment audio files. He also uses the open-source FFMPEG software to convert the lecture audio files to WAV format, as well as Adobe Audition CS6 to segment audio files into appropriate classes [17]. Because ASC is the heart of both his project and this thesis, these three tools could prove useful in this thesis' implementation.

Diving deeper into Govender's methodology, he uses one dataset composed of 150 lecture audio files in WAV format. This dataset is further subdivided into training and testing datasets. He creates it manually by segmenting the downloaded audio into speech and nonspeech snippets, resulting in 6,682 total audio files. To classify this audio, Govender implements a SVM classification model offered by the pyAudioAnalysis library. Using this library, Govender splits the lecture audio signal into homogenous pieces before applying the classification model to each of these segments. Since adjacent segments of the same class are morphed into a larger segment, this eventually produces an alternating sequence of speech and non-speech class labels for the audio file. He then writes a Python script to process these class labels, identifying the point where the Professor begins speaking, and the point where they finish speaking, thus producing the end points for trimming [17].

To evaluate the SVM classifier's performance, Govender subdivided this dataset into ten subsets, performing 10-fold cross-validation by training the SVM model with nine subsets and testing it with one subset, ten times over. Thus, every subset is a training set nine times and a testing set once [17]. *Figure 6* below shows the structure of the confusion matrix generated by the correct and incorrect cross-validation instance predictions.

	Actual speech	Actual non-speech
Predicted speech	TP	FP
Predicted non-speech	FN	TN

Fig. 6: Confusion Matrix for a Speech Discrimination Model. Source: [17]

Speech files correctly predicted as speech by the model are true positives (TP). Similarly, non-speech files correctly predicted as non-speech are true negatives (TN). These are the two desired outcomes. Contrarily, speech files predicted as non-speech by the model are false negatives (FN), and non-speech files predicted as speech are false positives (FP). The smaller the amount of false positives and negatives, the more accurate the classifier is [17].

The results from Govender's 10-fold cross-validation procedure are compiled into *Figure 7.* Note that this figure is the same as *Figure 6,* except for having an experimentally calculated value for each matrix entry. After training and testing the SVM classifier, he obtained 3,376 true positives, 3,342 true negatives, 44 false positives, and 100 false negatives.

	Actual speech	Actual non-speech
Predicted speech	3376	44
Predicted non-speech	100	3342

Fig. 7: Confusion Matrix to Evaluate SVM Classifier Performance. Source: [17]

Using these results, he calculates four performance metrics to evaluate the classifier's performance. The first metric, precision, is the ratio of correctly predicted speech audio to all the audio files that are predicted as speech [17]. He calculates precision as follows:

$$Precision = \frac{TP}{TP + FP}$$

Fig. 8: Precision Formula. Source: [17]

This classifier yielded a precision measure of 98.7%, indicating that the amount of false positives produced by this classification model is low. For comparison, Govender cites another study with a precision measurement of 82.4% [17] [18].

The second metric Govender determines is recall, which is the probability that the classifier can correctly identify speech. To do so, he uses the following formula:

$$Recall = \frac{TP}{TP + FN}$$

Fig. 9: Recall Formula. Source: [17]

In other words, recall is the ratio of correctly identified speech to all speech identified by the classifier, which was 97.1% for this particular one [17].

Govender also calculates accuracy, which is the ratio of correctly predicted speech and non-speech. Accuracy can be misleading when the class imbalance is large, and is hence often used with other metrics [17]. It is computed as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Fig. 10: Accuracy Formula. Source: [17]

The accuracy for his classifier was 97.9%, surmounting the previous SVM classifier studies' accuracy of 85.5% [17][18].

The last metric Govender calculates is F-measure, which finds the harmonic mean of precision and recall to evaluate the classifier's speech prediction [17]. F-measure can be a more preferred metric than accuracy, as it can handle imbalances in dataset size. It is calculated as:

$$F - measure = \frac{2\,(precision \times recall)}{precision + recall}$$

Fig. 11: F-Measure Formula. Source: [17].

Govender computed an F-Measure of 97.9% for his classifier, which outperformed his reference studies' F-measure of 73.8% [19]. Due to the simplicity in calculating these metrics and their abundance in other studies on SVM classifiers, all four are excellent candidates for the evaluation of this thesis project.

On top of evaluating the SVM classifier's performance with these four metrics, Govender also assessed lecture audio trim point prediction accuracy. To do so, he compared the trim points identified using the *pyAudioAnalysis* library to the "gold standard" trim points, which were points selected by staff at the University of Cape Town who had to manually edit these lecture videos in the past [17]. *Figure 12* shows the deviation between trim points predicted using ML and the "gold standard" trim points.

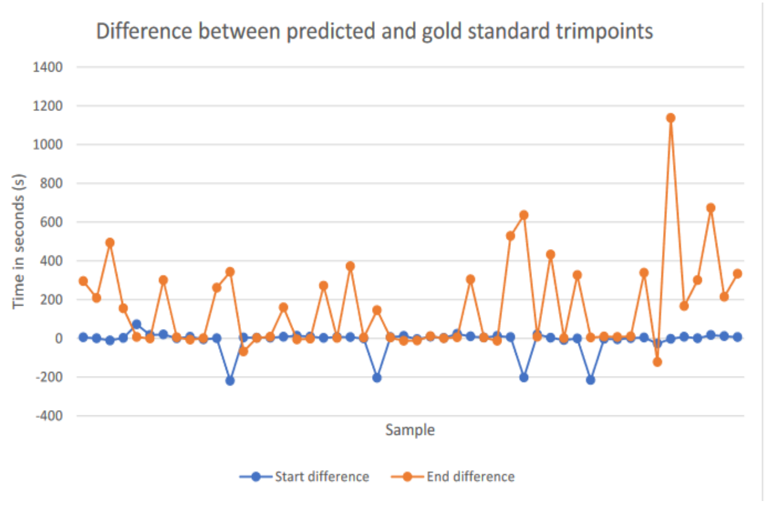


Fig. 12: Difference Between Predicted and Gold Standard Trim Points. Source: [17].

The blue line indicates the deviation between the start of lecture as determined by the classifier, and the gold standard start trim point. Similarly, the orange line depicts the difference between the predicted and gold standard lecture end trimpoint. The closer each point is to zero, the less time discrepancy there is between prediction and gold standard, making that prediction more accurate. The mean start trim point discrepancy was -11.22 seconds, and the mean end trim point discrepancy was 145.16 seconds [17].

Govender highlights how 90% of lecture start points were within 30 seconds of the gold standard data, which is fairly acceptable for the amount of time between when the video begins and when a professor starts talking. Conversely, only 50% of predicted lecture end points were within 30 seconds of the gold standard data [17]. He attributed this error to a high amount of background noise, as well as the lecture hall microphones not projecting a dominant voice properly. More specifically, if students were having a conversation after the lecture had ended and the classifier identified their conversation as dominant speech, it would induce significant error in lecture end point prediction [17].

vi. Conclusion

The papers discussed sections i through iv of this review vary greatly in their scope, despite all of them implementing machine learning concepts and describing processes of signal analysis. A more concrete next step could be integrating the fundamentals of each paper into an actual ML model for processing audio signals. For example, a ML algorithm could be written to scrub unwanted sounds from a microphone recording while keeping semantic sounds. In the context of an instructional YouTube video or lecture recording, breaths, throat clears, and coughs could be erased from the audio file, leaving behind only English words with meaning. Although this does not seem like a tremendous change, the resulting recording would be far more succinct.

Such a project would incorporate the fundamental feature selection criteria of [12] and [15], the feature extraction mathematics from [9], [11], [13], and [16], and the robust dataset criterion of [10] and [14]. The features extracted from this data set of instructional video audio could use MCME as a primary feature, given its promising performance in Kim et al.'s study on speech discrimination [13]. The resulting model would use concepts from each paper for a novel application that was minimally referenced in any paper besides Govenders project on lecture audio classification in [17].

Likewise, this thesis owes Govender's paper a tremendous amount of appreciation, as his project served as the basis for the methods followed and results obtained in this one. Similar to Govender's thesis, an SVM with a linear kernel was chosen to classify the audio examined in this thesis. Moreover, several of the performance metrics, as well as the comparison to gold standard data, were applied to the evaluation of results in this thesis, after observing them in Govender's. The fundamental difference with Govender's ML model is that he trained it to trim the unwanted start and end points of a lecture recording [17]. These are the portions of the lecture before the lecturer starts talking, and after he has finished talking, and there are similarly only two segments to be trimmed out per video. On the contrary, this thesis aimed to trim the hundreds of segments in between semantic, spoken words that are unwanted, such as breaths or lip smacks.

III. Methodology

i. Data Acquisition

As Kawahara et al. state in their paper on the automatic transcription of lecture speech, a ML model that accurately transcribes one speaker's lectures may perform horribly with a different speaker, due to the strong variation between speech patterns in individuals [14]. Likewise, each ML model was trained with the lecture audio data from one lecturer. I emailed Professor Steve Brunton and mathematics tutor PatrickJMT for permission to use the hundreds of lecture videos they have available on their YouTube channels. They both agreed, provided that the data obtained from these videos is not repurposed for any kind of commercial use, and is used solely for this thesis. The audio within these videos is ideal for training each ML model because the respective lecturer is the only person talking, the videos are untrimmed, and there are several hundred hours of it available on each YouTube channel. The Y2DOWN YouTube to WAV converter was used to isolate the audio from these lecture videos into a WAV file format, which was the ideal type of data for the Python libraries used to generate the ML models.

ii. Data Classification Taxonomy

After downloading these lecture audio files, the dataset for each lecturer was created by subdividing each file into hundreds of sound segments containing one sound from the taxonomy shown in *Figure 13* below. The Audacity sound editing software was used to segment these files. If a sound such as "exhaling" was detected, for example, it would be clipped off into its own, roughly two second long WAV file and labeled with the “nonspeech” class label. This resulted in a single folder containing many manually classified sound files for each lecture video.

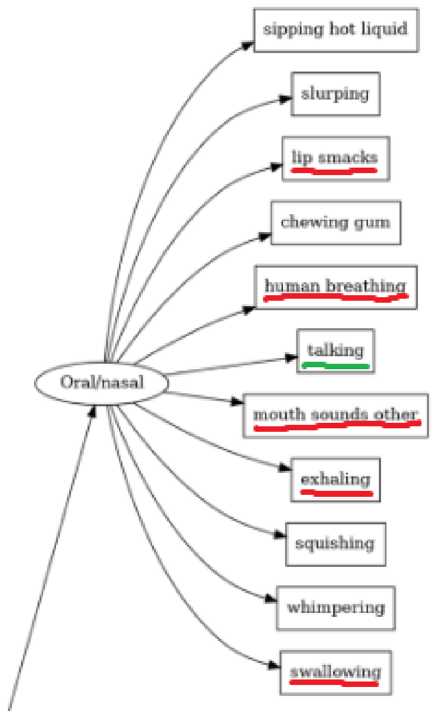


Fig. 13: Human Sound Classification Taxonomy. Source: [20].

Figure 13 depicts the human sound classes that each audio file was labeled with. The class underlined in green, talking, describes spoken words conveying useful information, which are the sounds that should be kept in the lecture recording. All other sound classes, underlined in red, describe noises that the ML models were trained to identify for removal from the lecture recordings. The six classes used in this thesis were adapted from the sound taxonomy used in D. Benesch et al. 's FOAMS soundbank [20]. FOAMS is an open source misophonia stimuli dataset; since this thesis wanted to explore ML models personalized to a single lecturer, it opted to create its own dataset based on the FOAMS taxonomy, rather than using the FOAMS dataset itself.

The five class labels highlighted in red in *Figure 13* were combined into a singular nonspeech class label. So anytime a lip smack, breath, exhalation, swallow, or any other type of human mouth sound was encountered, it would be labeled as nonspeech. The speech class label, on the other hand, entails semantic sounds only. This binary classification scheme is important because the SVM model used in this thesis only needs to understand the difference between actual words and unwanted sounds; forcing the SVM to classify with six classes would only overcomplicate the process, especially when only two classes were truly needed.

<u>iii. Data Segmentation</u>

The Audacity audio editing software was used to segment the lecture recordings into the individual WAV files used to train the ML model. The full-length WAV file from each desired lecture was first downloaded from the internet and loaded into Audacity.

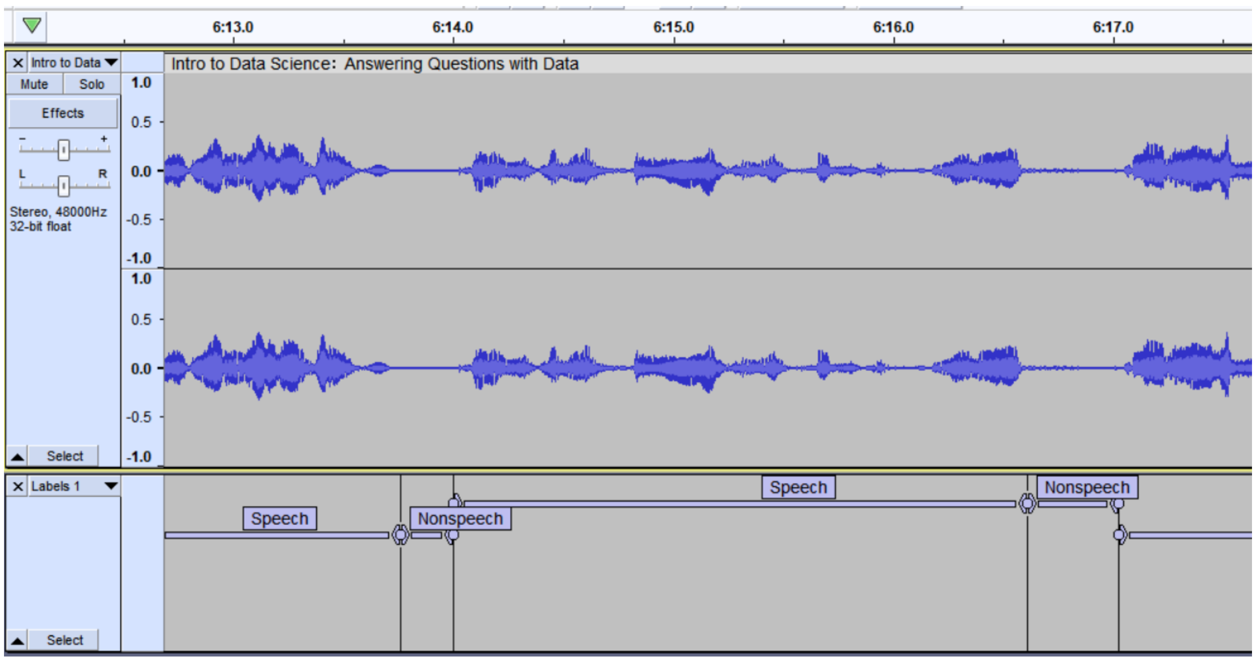


Fig. 14: Audio File Segmentation in Audacity Sound Editing Software

As shown in *Figure 14,* this file was then split into alternating segments of speech and nonspeech, and an appropriate label was assigned to each segment. Speech segments could vary anywhere from less than a second to tens of seconds in length, while nonspeech segments were typically not longer than a few hundred milliseconds. Once the entire lecture was partitioned, each segment was extracted as an individual WAV file. A single lecture typically yielded anywhere from 90 to 170 files for speech and nonspeech clips, each. These files were grouped by class before being stored in the same directory as the pyAudioAnalysis library.

iv. Classification Model

Both Rong et al. and Guo et al. explore the performance of audio classification algorithms using support vector machines (SVM) [9] [11]. Similarly, Govender's lecture audio classification model used a SVM with a linear kernel, which he claimed was less computationally demanding than a k-NN model, and performed better when a training dataset was available [17]. Building off the implementation of these studies, a SVM was used for the lecture audio classification model in this thesis. To do this, the pyAudioAnalysis library was used because it offers an extensive functionality, such as identifying the optimal classifier parameter for SVMs and extracting a diverse range of audio features [17].

v. Audio Features

The *pyAudioAnalysis* extracts 34 short-term features in total for use in its classification and segmentation functionalities [21]. This library was chosen because its accompanying default audio features perform well in speech and nonspeech discrimination [17]. Each feature is summarized in *Table 1* below.

Feature ID	Feature Name	Description
1	Zero Crossing Rate	Amount of sign changes of a signal within a given frame
2	Energy	Sum of squares of signal values, normalized by the corresponding frame length
3	Entropy of Energy	The amount of sudden changes in an audio signal
4	Spectral Centroid	The spectrum's center of gravity
5	Spectral Spread	The spectrum's second central moment
6	Spectral Entropy	The entropy of normalized spectral energies for some given sub-frames
7	Spectral Flux	The difference squared between the normalized magnitudes of the spectra of two given frames
8	Spectral Roloff	The frequency below which 90% of the spectrum's magnitude distribution is located
9-21	MFCCs	Provides a cepstral representation where the frequency bands are distributed on a mel-scale rather than a linear scale
22-23	Chroma Vector	A representation of spectral energy divided into 12 bins, where each bin represents one of the 12 pitch classes of

		western-style music
34	Chroma Deviation	Standard deviation of the above chroma coefficients

Table 1: Audio Features Used for Extraction by pyAudioAnalyis. Source: [21]

vi. Evaluation of Research Results

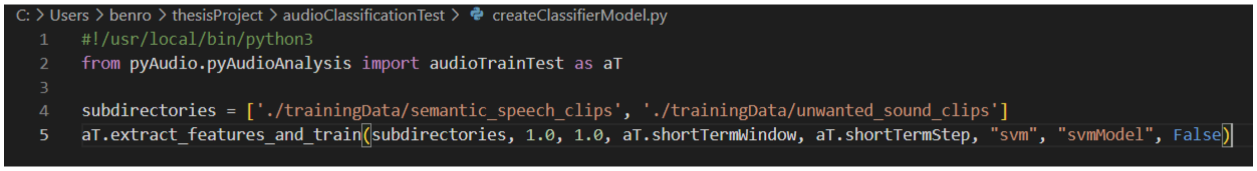
Inspired by Govender's thorough evaluation of results in [17], this thesis performed two evaluation stages in a similar fashion. First, the performance of the SVM classifier was determined using a 10-fold cross-validation procedure. A subset of the lecture audio data was split into ten equally sized portions before training the SVM classifier with nine portions and testing it with one portion, repeating for a total of ten times. A confusion matrix exactly like that in *Figure 6* was generated from this procedure, and used to calculate precision, accuracy, recall, and F-measure. These four performance metrics were then compared with other recent papers that evaluated the same metrics for perspective.

The other evaluation procedure that was borrowed from Govender's methodology was the comparison of computer determined lecture trim points to "gold standard data" [17]. The start and end timestamps for each unwanted sound in the audio for a lecture file was determined by a python script that incorporated the SVM classification model. That is, after the classifier identified each wanted and unwanted sound segment in the lecture audio file, this python script would scan the length of that audio file, marking these timestamps as it went along. This same procedure was done manually for five selected lectures, creating the gold standard data. Finally, the discrepancy between human-determined and computer-determined trim points was calculated, in order to determine the percentage of lecture audio that was accurately partitioned.

IV. Report and Discussion of Research Results

i. Sound Classification with User-Trimmed Audio Clips

Throughout this project, several iterations of both datasets and SVM models were used as more was learned about how the model interacted with different dataset sizes and classification schemes. The very first model was created through supervised learning; lectures 1 through 4 from Steve Brunton's *Intro to Data Science* Youtube series were used to create a training dataset of 443 speech files and 439 nonspeech files. These files were segmented not at fixed intervals, but manually, based on the length of each sound belonging to either the speech or nonspeech class. For example, a segment of uninterrupted words spoken by Professor Brunton would be labeled as a semantic speech segment, with a length of a few seconds. If he took an audible breath or smacked his lips immediately after, that noise would be isolated into its own segment and assigned with the unwanted sound segment classifier. Likewise, each lecture WAV file was partitioned into alternating segments of speech and nonspeech labels of varying lengths.


C: > Users > benro > thesisProject > audioClassificationTest > createClassifierModel.py
```python
#!/usr/local/bin/python3
from pyAudio.pyAudioAnalysis import audioTrainTest as aT

subdirectories = ['./trainingData/semantic_speech_clips', './trainingData/unwanted_sound_clips']
aT.extract_features_and_train(subdirectories, 1.0, 1.0, aT.shortTermWindow, aT.shortTermStep, "svm", "svmModel", False)
```

Fig. 15: Python Script to Create SVM Classifier Model using *pyAudioAnalysis*

A simple python script incorporating the pyAudioAnalysis library was then written to train an SVM with a linear kernel using this data, as shown in *Figure 14* above. A short-term window and short-term step were used to train this classifier because of how "short-term" spoken words are in nature [21].

The first testing dataset was created with the same methodology described above. It contained 174 speech files and 173 nonspeech files, collected from lecture 5 of the same series. The SVM was then tested in its ability to correctly classify these files using the classifyFolder command line function from the *pyAudioAnalysis* library. From the semantic speech portion of the testing dataset, the model correctly identified 173 out of 174 files as speech, and incorrectly identified 1 file as nonspeech. From the unwanted sounds portion of the dataset, the model correctly identified 171 out of 173 files as nonspeech, and incorrectly identified 2 out of 173 files as speech. The results are summarized in *Table 2* below.

	Correctly Identified	**Incorrectly Identified**	**Total**
Speech Files	173	1	174
Nonspeech Files	171	2	173

Table 2: Audio File Classification Success Rate for First SVM

It is important to note that the audio files in the testing dataset were manually segmented. Although the data is reassuring at first glance, the ML model would not encounter these perfectly partitioned audio segments in the real world, but instead would work with segments of the same exact length, such as one fourth of a second. Likewise, this thesis strives to create a model that can classify wanted and unwanted sounds given an unsegmented lecture audio file as input. The next section in this discussion highlights a refined SVM's ability to classify sounds given an unsegmented audio file that is partitioned into fixed size segments.

ii. Sound Classification with Fixed-Size Audio Clips

Unlike the testing dataset from above, the second dataset was created by creating and classifying fixed-size labels of one second in length on lecture 6 from the same series. This is more realistic, as any given lecture audio file can be partitioned into these segments before classifying each segment with the ML model. In this iteration, the training dataset contained 618 speech files and 613 nonspeech files. The testing dataset contained 334 speech files and 143 nonspeech files. All the audio files in the training dataset were of varying length, while each file in the testing dataset was exactly one second in length. As shown in *Table 2* below, 326 out of 334, or 98% of the semantic speech audio files were correctly classified as such by this SVM.

	Correctly Identified	**Incorrectly Identified**	**Total**
Speech Files	326	8	334
Nonspeech Files	40	103	143

Table 3: Audio File Classification Success Rate for Second SVM

Despite this success, the model struggled with nonspeech audio files, correctly classifying only 40 out of 143, or 30% of the given files. Upon further inspection, the majority of incorrectly identified nonspeech files contained the filler words "uhh" or "umm".

This makes sense considering how the four main sounds that were classified as unwanted, nonspeech sounds were lip smacks, breaths, and these two nonlinguistic words. The sounds produced by "uhh" and "umm" are encountered within countless semantic words, such as "bump" and "freedom" to name a few. Lip smacks and inhalations, on the other hand, are not subsets of any word in the English language. This justifies the poor performance of the model on nonspeech files; many files containing only lip smacks and inhales were correctly classified, whereas files containing "uhh" and "umm" confused the model. The next section highlights how the training data was reclassified in light of this consideration.

iii. Performance Analysis after Altering Classification Scheme

To alleviate the poor classification performance of the SVM on nonspeech files, a design sacrifice was made to relabel “uhh” and “umm” sound segments as semantic speech. Likewise, the only three noises that would be classified as unwanted sounds with this new scheme would be lip smacks, inhalations, and swallows. All five lectures used to construct the training dataset were revisited, combining each of these relabeled sounds with the already existing semantic sound segments adjacent to them. The new SVM model was tested on fixed-size, pre-classified speech and nonspeech files of exactly one second in length.

	Correctly Identified	**Incorrectly Identified**	**Total**
Speech Files	356	3	359
Nonspeech Files	14	105	119

Table 4: Audio File Classification Success Rate with 1 s Label Window

Comparing *Table 3,* where “uhh” and “umm” were classified as unwanted sounds to *Table 4,* where these noises were reclassified as speech, we can see an improvement in correctly identified speech files from 326 out of 334, or 97.6%, to 356/359, or 99.2%. Interestingly, however, the amount of correctly identified nonspeech files actually decreased, dropping from 40/143, or 30%, to 14/119, or 11.8%. Counterintuitive to the justification above, the amount of correctly identified nonspeech files actually decreased. Despite this, the new classification scheme of identifying only lip smacks and breaths was kept, as the removal of “uhh” and “umm” from lecture speech could leave too small of a gap between adjacent words, causing annoying clashes between words in the processed speech file.

Classifying these one second lecture audio segments for testing was also intrinsically problematic, as a one second window typically cannot separate speech and nonspeech sounds adequately. For example, a one second segment could contain both speech, such as the word “apple” and nonspeech, like an inhalation, making its classification challenging. Likewise, a second round of fixed-size audio segment testing was conducted, but with 0.25 second segments instead of one second segments. The results of this change are summarized in *Table 5* below.

	Correctly Identified	**Incorrectly Identified**	**Total**
Speech Files	514	97	611
Nonspeech Files	103	7	110

Table 5: Audio File Classification Success Rate with 0.25 s Label Window

Most interestingly, the success rate for nonspeech file classification drastically increased to 103/110, or 93.6%. This is tremendously larger than the 11.8% success rate that was experienced with one second labels. The speech file classification success rate decreased to 514/611, or 84.1%. There were, however, considerably more speech files in this iteration. The previous speech file classification success rate of 99.2% was for 252 less audio files.

iv. 10-fold Cross Validation and Performance Metrics

The SVM classifier's performance was examined further by performing a 10-fold cross-validation procedure, similar to the one used in [17]. A total of 1,089 audio segment files classified as either speech or nonspeech were subdivided into 10 groups. Then, across 10 iterations, a new SVM model was trained with 9 of the groups and tested with 1 of them. Each test was done using the classifyFolder command line functionality from the *pyAudioAnalysis* library. The results are summarized in *Table 5* below.

	Actual Speech	**Actual Nonspeech**
Predicted Speech	545	2

Predicted Nonspeech	3	539

Table 5: Confusion Matrix for 10-Fold Cross-Validation Experiment

Four performance metrics were calculated using the results from this experiment, as Govender did in [17]. The first of which was precision, which is the ratio of correctly predicted speech files (545) to all of the files identified as speech (547). In other words, the precision of this classifier was 99.6%. Next, the recall of the classifier was computed. Recall is the total number of correctly predicted speech files (545) to all of the speech files in the dataset (548) [17]. This produced a recall of 99.5% for this classifier. The third calculated metric was accuracy, which was the ratio of correctly identified speech and nonspeech files (1,084) to the total amount of files used in the procedure (1,089). These values yielded an overall accuracy of 99.5%. Lastly, the F-measure was calculated, which is a metric with a more complicated formula that depicts how well the model can truly identify and classify speech [17]. The F-measure obtained in this experiment was 1.982/1.991, or 99.5%.

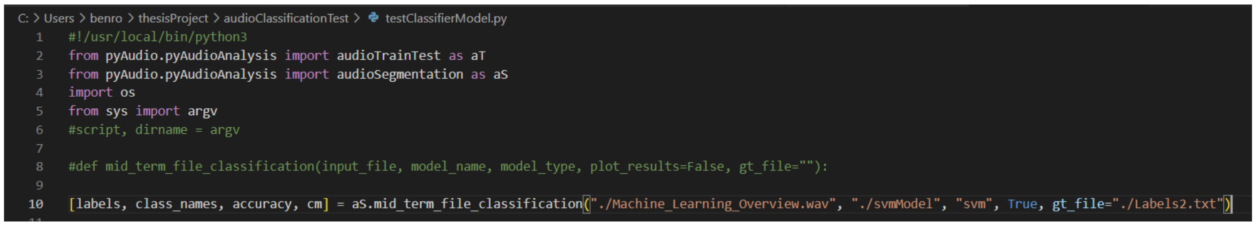
	This Thesis	**Govender [17]**	**Giannakopoulos, et al. [18]**	**Siantikos, et al. [19]**
Precision	99.6%	98.7%	82.4%	-
Recall	99.5%	97.1%	90.5%	-
Accuracy	99.5%	97.9%	85.5%	-
F-measure	99.5%	97.9%	-	73.8%

Table 6: Comparison of Cross-Validation Performance Metrics with Other Studies

Following this table, one would expect a justification raving about the "incredible performance" of the classifier tested in this experiment. This explanation is quite the opposite, however. These numbers obtained in this thesis are far too high, and for two main reasons. The first being that all the audio files used in this procedure must be custom sized label segments since the same data is being partitioned into different combinations of training and testing sets. In a future study, the SVM could be trained with labeled audio files of a fixed size; however, the bulk of data used in this thesis were custom sized segments, so it made sense to carry out the 10-fold cross-validation procedure with the same types of audio files. The second reason these metrics are uncannily high is because there were only 1,089 files used in this procedure. Govender, for example, performed his cross-validation test with 6,862 files, which is over six times as many files. All in all, although the performance metrics from this procedure are slightly reassuring, they are mostly deceiving, and they don't paint nearly as clear a picture of the true efficacy of the model as evaluating accurate trim point identification does, which is outlined in the next section.

v. Evaluating Trim Point Detection

A total of 179 seconds from Lecture 6 were segmented into .25 second segments and labeled as either speech or nonspeech according to the classification scheme. The SVM was then evaluated in its ability to produce accurate speech and nonspeech timestamps using the python script in *Figure 16*. Overall, the SVM correctly identified 161.5 seconds of audio into its appropriate categories, or roughly 90% of the lecture audio. Moreover, most of the incorrectly identified segments were at the transition point between speech and nonspeech. This is important to note because the majority of audio segments were correctly classified; only the outskirts of a given pattern of adjacent nonspeech segments, for example, would cause trouble.


C: > Users > benro > thesisProject > audioClassificationTest > testClassifierModel.py

```python
#!/usr/local/bin/python3
from pyAudio.pyAudioAnalysis import audioTrainTest as aT
from pyAudio.pyAudioAnalysis import audioSegmentation as aS
import os
from sys import argv
#script, dirname = argv

#def mid_term_file_classification(input_file, model_name, model_type, plot_results=False, gt_file=""):

[labels, class_names, accuracy, cm] = aS.mid_term_file_classification("./Machine_Learning_Overview.wav", "./svmModel", "svm", True, gt_file="./Labels2.txt")
```

Fig. 16: Python Script to Test Lecture Trim Point Identification with *pyAudioAnalysis*

Table 6 elaborates on the types of errors that were encountered in the model's trim point calculations. Each of these errors arose from an incorrectly identified, 0.25 second audio clip that was segmented from this lecture. Twenty five of these errors occurred in a speech to nonspeech transition; in other words, the error took place in a clip that had a correctly identified speech segment immediately before it. Conversely, only one error occurred in a nonspeech to speech transition, where the segment immediately before this incorrectly classified segment was a correctly classified nonspeech segment. Pure speech is a segment that is sandwiched in between speech clips on both sides. No errors occurred within these types of segments. On the other hand, pure nonspeech clips, which are adjacent to nonspeech segments on both sides, were misclassified 47 times. For the 716 total 0.25 second segments tested, 73 segments observed errors.

Type of audio clip	**Speech to nonspeech transition**	**Nonspeech to speech transition**	**Pure speech**	**Pure Nonspeech**
Occurrences	25	1	0	47

Table 7: Types and Amounts of SVM Trim Point Detection Error

This 90% accuracy in identifying audio trim points is the most important takeaway of the entire results section. Most of the results highlighted in previous sections were reassuring at best, and did not paint nearly as clear of a picture of the classifier's efficacy in its intended application. While 90% is a worthwhile accomplishment for the first time training a classifier to remove sounds that would bother someone with misophonia, future works would want to push this as close to 100% as possible. This would make the edited lecture video more concise for any viewer, and especially a better listening experience for those with misophonia.

V. Conclusion and Implications for Further Research

i. Overview

After reviewing the contemporary research on ML and its applications for audio signal processing in *section II*, this thesis began its own exploration in the field of machine learning for audio processing by creating an audio dataset tailored specifically to one lecturer's speech patterns. Hundreds of audio clips ranging from milliseconds to seconds in length were isolated from Professor Steve Brunton's lecture recordings, before labeling each segment as "speech" or "nonspeech" in the Audacity audio editing software. These labeling decisions were made based on the sound classification scheme described by Benesch et al. in [20]. Speech segments were any uninterrupted string of spoken words containing meaning. Nonspeech segments, on the other hand, were clips containing non-semantic, speaker made noises, such as lip smacks, inhalations, and throat clears. The isolated, labeled audio files produced by Audacity's label creation functionality were used to train a SVM classifier with the *pyAudioAnalysis* open-source Python library.

The SVM's ability to differentiate between speech and nonspeech clips it had not seen before was assessed through the comparison with pre-labeled testing data, as well as by performing a 10-fold cross-validation. The segmentation functions from *pyAudioAnalysis* were also used to generate lecture audio trim points using this SVM classifier, before comparing with trim points that were generated for the same lecture by hand, through listening to the audio and manually marking unwanted segments. Overall, the SVM trained in this thesis obtained 90% accuracy in correctly identifying trim points for unwanted sounds in lecture audio. This was similar to Govender's results in [17], where 90% of his SVM model's predicted lecture trim points were within 30 seconds of the gold standard data.

ii. Answers to Scientific Questions

Two research questions were posed in *section I* of this thesis. The first question asked how the automated trimming of lecture videos could be reshaped for people with misophonia, while having the result provide a clean and concise lecture watching experience for everyone, regardless of their aversion to the sounds that bother people with misophonia. The answer to this question began with the classification scheme developed in *section III*, and was refined after the examination of the initial research results in *section IV.* While the original objective was to filter out the sounds "uhh", "umm", lip smacks, inhalations, and swallowing, the final SVM classifier was trained only to eliminate the last three sounds. Since the first two are fundamental sounds within other meaningful words, and are often too close to adjacent words in time to be cleanly trimmed out, they were labeled as meaningful speech when creating the final training dataset. This design choice crafted a succinct lecture watching experience for typical students, while still identifying a few key trigger sounds which, after their removal from the lecture audio, would greatly benefit someone with misophonia's experience as well.

This thesis also asked how to test a ML model's ability to support the removal of unwanted sound segments from a lecture recording. To accomplish this, the ML model trained in this thesis underwent basic file classification tests to quantify how many unseen audio files could be correctly identified by the SVM. Additionally, more complicated tests were implemented, such as the 10-fold cross-validation procedure as seen in *section IV*. Moreover, a few lectures were manually listened to, before using the Audacity audio editing software to create fixed-size class labels of either “speech” or “nonspeech” and correspondingly classify these lectures by hand. Next, the SVM classifier was given the same task, and the trim points it identified were compared with those generated by the human ear in the previous step.

iii. Implications for Future Research

Going off the idea of making the overall lecture audio trimming process described in this thesis more automated, two more functionalities could be added to the existing project. First of all, although the model could adequately highlight portions of audio that need to be trimmed out with 90% accuracy, some portions that were missed by the model would still need to be manually trimmed. This was due to the lower than expected accuracy in comparison with gold standard data. This is a promising first step in the search for a model that can cut out these portions with high accuracy in an unsupervised way, however, more data needs to be added in order to adequately achieve it with minimal error. Thus, a future study could implement the automatic editing of a lecture video, given the trim points generated by the SVM classifier. A python script could be written incorporating the MoviePy library, which has functionalities for removing audio and video segments for given timestamps [22]. This avenue was not explored in this thesis, however, as it would be beneficial to improve the 90% trim point classification before feeding the trim point data to a video editing script like this.

On the opposite end of the system functionality, implementing unsupervised learning would make the system much less burdensome to implement into a university’s lecture recording and publication framework. In this thesis, the training dataset for the SVM model that classified between wanted and unwanted audio clips was created by manually listening to lecture audio segments, before classifying them as either speech or nonspeech. On one hand, once an adequate training dataset is produced for a single lecturer, its resulting model can be used to edit a lifetime of that professor's lecture videos. On the other hand, it would be far easier if a general dataset could be applied to training on different lecturer's audio, therefore removing the manually intensive part of this system.

VI. Bibliography

[1] J. Brout, M. Edelstein, M. Erfanian, M. Mannino, L. Miller, R. Rouw, S. Kumar, and M. Rosenthal, "Investigating Misophonia: A Review of the Empirical Literature, Clinical Implications, and a Research Agenda," *frontiers in Neuroscience,* Feb. 7, 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5808324/

[2] "Automatic Video Editor." kapwing.com. https://www.kapwing.com/tools/smart-cut (accessed April 28, 2023).

[3] "Silence Remover." podcastle.ai. https://podcastle.ai/products/silence-removal (accessed April 28, 2023).

[4] S. Brunton. "ME565 Lecture 16: Discrete Fourier Transforms (DFT)," *Youtube,* Apr. 27, 2016. [Video file]. Available: https://www.youtube.com/watch?v=KTj1YgeN2sY&t=18s

[5] 3Blue1Brown. "But what is the Fourier Transform? A visual introduction," *Youtube,* Jan. 26, 2018. [Video file]. Available: https://www.youtube.com/watch?v=spUNpyF58BY&t=740s

[6] "Convert MP3 to WAV," *pythonbasics.org,* 2021. [Online]. Available: https://pythonbasics.org/convert-mp3-to-wav/

[7] J. Echessa, "How to trim a video using FFmpeg," *shotstack.io,* Jun. 27, 2021. [Online]. Available: https://shotstack.io/learn/use-ffmpeg-to-trim-video/

[8] R. Park, "Spotify may know you better than you realize," *Digital Information and Transformation, MBA Student Perspectives,* Harvard University, Apr. 8, 2018. [Online]. Available: https://digital.hbs.edu/platform-digit/submission/spotify-may-know-you-better-than-you-realize/.

[9] F. Rong, "Audio Classification Method Based on Machine Learning," in *Proc. of the 2016 Int. Conf. on Intelligent Transportation, Big Data & Smart City, ICITBS 2016, 17-18 Dec 2016, Changsha, China* [Online]. Available: IEEE Xplore, https://ieeexplore.ieee.org/document/8047110.

[10] H. Homburg, I. Mierswa, B. Möller, K. Morik, and M. Wurst, “A Benchmark Dataset for Audio Classification and Clustering,” in *Proc. of the 2005 6th Int. Conf. on Music Information Retrieval, ISMIR 2005, 11-15 Sept. 2005, London, United Kingdom* [Online]. Available: Research Gate, https://www.researchgate.net/publication/220723598_A_Benchmark_Dataset_for_Audio_Classification_and_Clustering.

[11] G. Guo and S. Z. Li, “Content-based audio classification and retrieval by support vector machines,” *IEEE Transactions on Neural Networks,* vol. 14, no. 1, Jan. 2003. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1176140/.

[12] B. Logan, “Mel Frequency Cepstral Coefficients for Music Modeling,” in *Proc. of the 2000 1st Int. Symposium on Music Information Retrieval, ISMIR 2000, 23-25 Oct. 2000, Plymouth, Massachusetts* [Online]. Available: ResearchGate, https://www.researchgate.net/publication/2552483_Mel_Frequency_Cepstral_Coefficients_for_Music_Modeling.

[13] B. Kim, D. Choi and Y. Lee, “Speech/Music Discrimination Using Mel-Cepstrum Modulation Energy,” in *Proc. of the 10th Int. Conf. on Text, Speech and Dialogue, TSD 2007, 3-7 Sept. 2007, Pilsen, Czech Republic* [Online]. Available: SpringerLink, https://link.springer.com/book/10.1007/978-3-540-74628-7.

[14] T. Kawahara, H. Nanjo and S. Furui, *“*Automatic transcription of spontaneous lecture speech*,” in Proc. of the 2001 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2001, 9-13 Dec. 2001, Madonna di Campiglio, Italy* [Online]. Available: IEEE Xplore, https://ieeexplore.ieee.org/document/1034618.

[15] G, Sharma, K. Umapathy, and S. Krishnan, “Trends in audio signal feature extraction methods,” *Journal of Applied Acoustics,* vol. 158, January 2020. [Online]. Available: Elsevier ScienceDirect, https://www.sciencedirect.com/science/article/pii/S0003682X19308795.

[16] G. Wichern and J. Le Roux, “Phase Reconstruction with Learned Time-Frequency Representations for Single-Channel Speech Separation,” *in Proc. of the 2018 16th Int. Workshop on Acoustic Signal Enhancement, IWAENC 2018, 17-20 Sept. 2018, Tokyo, Japan* [Online]. Available: IEEE Xplore, https://ieeexplore.ieee.org/document/8521243?denied=.

[17] D. Govender, “Investigating audio classification to automate the trimming of recorded lectures,” M.S. Thesis, Sch. of Information Technology, Univ. of Cape Town, 2018. [Online]. Available: https://pubs.cs.uct.ac.za/id/eprint/1260/1/Thesis-final.pdf

[18] T. Giannakopoulos, D. Kosmopooulos, A. Aristidou, and S. Theodoridis, “Violence content classification using audio features,” *Advances in Artificial Intelligence*, vol. 3955, pp. 502-507,

2006. Accessed Feb. 26, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/11752912_55

[19] G. Siantikos, T. Giannakopoulos, and S. Konstantopoulos, (21-22 Apr. 2016). Monitoring Activities of Daily Living Using Audio Analysis and Raspberry PI: A Use Case on Bathroom Activity Monitoring. Presented at the 2nd Int. Conf. on Information and Communication Technologies for Ageing Well and eHealth (ICT4AWE), Rome, Italy. [Online]. Available: https://www.iit.demokritos.gr/sites/default/files/9783319627038-c2.pdf

[20] D. Benesch, D. Orloff, and H. Hansen, “FOAMS: Processed Audio Files.” Zenodo.org. https://zenodo.org/record/7109069#.ZA0FhB_MJPY (Accessed March 11, 2023).

[21] T. Giannakopoulos, “pyAudioAnalysis Wiki.” github.com/tyriannak https://github.com/tyiannak/pyAudioAnalysis/wiki/5.-Segmentation (accessed April 28, 2023).

[22] Zulko, “MoviePy User Guide.” zulko.github.io/moviepy https://zulko.github.io/moviepy/ (accessed April 28, 2023).