Kevin Yu, Ethan Marcello, and Nicholas Guidoboni
EECE 5666 - Digital Signal Processing
Professor Mohammad Tajdini
April 28, 2021

Digital Processing of Audio Signals

I. Introduction

As audio processing technology has continued to develop over recent years, one problem that engineers and researchers have been forced to tackle is the extraction of precise data and information from complicated and noisy signals. No matter what recording method is utilized, any desired audio signal will likely be accompanied by problems such as unwanted background noise and electrical interference. In addition, the accuracy of any received audio signal is also constrained by the ability of its sensing equipment--i.e. the microphone--to recover the original signal with perfect resolution. These combined issues have the potential to make an audio signal completely worthless to its user, as the desired data is lost behind a cloud of noise. In order to eliminate or reduce the impact of these undesired effects, the signal must be properly processed and filtered before implementation. In order to do this, the signal must be extracted, filtered, and processed through a well-developed algorithm. The details of this algorithm can vary greatly depending on the intended result of the audio signal and the limitations of the researchers involved. Variations between algorithms can include, but are not limited to, different filtering techniques, alternative extraction methods, software implementation, and sample size. While each individual audio processing algorithm comes with its own benefits and drawbacks, there are some methods that have been proven to be more successful than others for certain applications. In this paper, our team will focus on the research behind these notably successful algorithms and attempt to explain their implementation within the audio processing industry. In addition, we will design an experiment in which we utilize these techniques to create independently developed audio processing algorithms within MATLAB.

II. Problems and Applications in DSP

One of the most common problems in any signal processing field is noise, and in audio this is no exception. Noise is defined as any registered signal that degrades the quality and accuracy of any desired output from a given input. Within the audio realm, noise primarily affects recorded or live transmission of audio. Typical sources of noise can come from the electronic components themselves, audio multipath propagation, or external factors such as wind. There are solutions that help mitigate the effect of noise, whether that is passive technology (i.e. deadcat covers on microphones), or active technology (i.e. filters, signal editing, etc.).

In addition to the general case of noise, there are also specific problems such as source separation, which attempts to distinguish and filter out a certain audio source. The most common example where this is useful is the 'cocktail party problem', where the desired outcome is selective hearing of the person you are talking to. An application where this would be useful is determining who is speaking in a room when hosting a digital conference. Additionally, source separation can be useful when one wants to extract a certain sound from a recording (e.g. when sampling music), which can help with audio synthesis. One final application of source separation is for hearing aids, where it is useful to filter or amplify specific frequencies for individual users.

III. Problem Statement

Overall, the focus of our paper will be to analyze the principal features of several audio signal samples in order to detect certain speech patterns and eliminate the presence of background noise. Data will be collected in several different environments using a smartphone microphone. Although hardware limitations can have an effect on the signal recorded by the input transducer, this factor will be largely ignored in our analysis since these effects are negligible. The main focus will be on the filtering and processing of the digital signal that is obtained after the signal passes through the analog to digital converter. We will be making quantitative comparisons between the digital signals obtained to analyze the effects of our filtering techniques, and we will also note the qualitative results through listening to the original audio sample in comparison to the filtered one.

It will be important to first determine the main frequencies found in human vocalizations, and to suggest, choose, and implement audio signal filtering techniques. Audio signals will collect human speech and background noise (music, traffic, etc.), and the components of these signals are set. Using MATLAB, it is first proposed to take the fast fourier transform (FFT) of the signal to represent it in the frequency domain. We can then use a bandpass, high, or low pass filter to eliminate frequencies outside of the expected speech domain. Then, invert the signal back to the time domain and play the audio file in the absence of background noise. This can be compared to an audio sample taken in isolation and devoid of background noise to help quantitatively measure our results by inspecting the relative power of selected frequencies between the two signals. In addition, we will analyze more recent signal processing methods in the audio processing industry, such as voice activity detection algorithms and systems that incorporate a deep-learning approach. Those processes will be expanded upon in the following sections.

IV. Research and Processing Methods

Feature Extraction

One of the most important aspects of audio signal processing is feature extraction. In a broad sense, feature extraction refers to the isolation and analysis of compact representations of a given signal. Any feature extracted through this operation will be a more compact version of the original signal that highlights its key aspects and characteristics through a particular data set. A suitable feature will mimic the properties of a signal in such a way that it can be used for calculations within a minimal margin of error. This is the main advantage of feature extraction, as it produces much more manageable groups for processing information through an algorithm.

In audio processing, feature extraction can generally be separated into three main categories: time domain features, frequency domain features, and cepstral domain features. Typically speaking, time domain features are the easiest to process because they deal with the audio signal in its natural form. For the purposes of speech recognition, one key feature commonly extracted in voice detection algorithms is the zero-crossing rate (ZCR) of the signal. In a broad sense, the ZCR of an audio frame is defined as the rate of change of the sign of the signal during a specified interval. Mathematically speaking, "it is the number of times a signal changes its sign from positive to negative and vice versa, divided by the length of the frame" [1]. The reason the ZCR is a key feature is that it can be easily and efficiently utilized for voice activity detection, i.e., whether a given frame is voiced, unvoiced, or silent. R.G. Bachu et. Al. discuss this phenomenon in their paper entitled "Separation of Voiced and Unvoiced using Zero Crossing Rate and Energy of the Speech Signal." It has been experimentally proven that voiced speech displays a low zero-crossing count, while unvoiced speech results in a high zero-crossing count. During silence, the zero-crossing rate is theoretically zero, however, this value cannot realistically be achieved due to the presence of noise within the signal [2]. From these distinctions, basic speech recognition patterns can be built that classify sound into those three categories: voiced, unvoiced, and silent. In addition to the zero-crossing rate, other features can be extracted within the time domain. These include, but are not limited to, volume, short time energy, and temporal centroid. For the purposes of speech recognition, it can be valuable to analyze these parameters, however, it is typically more beneficial to extract features from the frequency and cepstral domains as they provide more appropriate data.

In order to extract features from the frequency domain, the signal must be converted from the time domain using either the Fourier transform or an auto-regression analysis. For the purposes of this paper, our team will focus on the features that result from the time-frequency conversion utilizing the Fast Fourier transform, since it is the most practical method for implementation within MATLAB. As Sharma et. Al. explain in their paper, the most important frequency domain features to be extracted during audio processing, specifically for speech recognition, include: peak frequency, cutoff frequency, power, sub-band energy ratio, group delay function, fundamental frequency, and spectral slope [1]. Many filtering methods for audio processing rely on feature extraction from the frequency domain as it provides the necessary input parameters in order to design a filter. Other algorithms, such as MATLAB's Voice Activity Detection Toolbox, utilize the frequency domain in order to extract the power of the signal. By analyzing the signal power over various points, it can be determined whether there is active speech present. These are just some examples of feature extraction in the frequency domain. In reality, there exists a vast number of important properties that can be analyzed after performing the Fast Fourier Transform. Unfortunately, in-depth explanations of each individual feature go beyond the scope of this paper, however, many of those aforementioned principles will be implemented and discussed throughout the remainder of this report. As such, mathematical equations describing each feature will appear as they become relevant to the topic at hand.

Finally, audio processing features can be extracted and analyzed within the cepstral domain. In general, a cepstrum can be obtained by taking the inverse Fourier transform of the logarithm of the spectrum of a signal. This process will result in the development of a complex, power, phase, and real cepstrum. For speech analysis, the power cepstrum is the most relevant for processing purposes [1]. For practical application, the cepstrum is typically represented on the mel scale, which is a scale of pitches that have been experimentally determined to be equidistant from each other. Utilizing this scale, the most important feature to be extracted from the cepstral domain is the mel frequency cepstral coefficients (MFCCs). In general, MFCCs represent the short-time power spectrum of an audio signal. The combination of these coefficients creates an "acoustic vector" which can be utilized to recognize individual voice characteristics of any given speaker [3]. This process has been successfully implemented into a wide variety of audio industries, such as speech recognition, speech enhancement, music genre classification, and vowel detection.

Digital Speech Models

______To analyze audio with source separation, we have to create models of how speech works. Generally, researchers present four different models for speech: source filter, linear prediction, sinusoidal, and harmonic/noise models. For this project, the first three models will be used as references. When researchers present the source filter mode, the assumption is that there is varying pressure in the larynx combined with the 'filter' of the vocal tract can simply be represented as a convolution, where s(n) is the speech signal, h(n) is the impulse response of the vocal tract, and u(n) is the excitations from the larynx.

$$s(n) = \sum_{m=0}^{\infty} h(m)u(n-m)$$

This equation can be represented as the following equation below in the Z-domain for analysis, where G is the gain of the system and p is the order of the system. This transfer function can also be expressed as a difference equation.

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} \mathrm{a}_k z^{-k}} \qquad s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n)$$

By observation, we can see that the form of this filter is very simple; however, depending on the complexity of the signal, it might not be the best choice for accuracy.

For the second model they present, they build upon the first model, however, they add an error signal in order to perform predictions on the unknown parameter a_k. To solve for this parameter, the minimum means squared error is used.

$$e(n) = s(n) - s_p(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k)$$

Similar to the source filter model, this approach is also relatively simple, but suffers from the same accuracy issues.

Finally, for the last model, speech is modelled as a sum of sinusoids with various frequencies and phases.

$$s(n) = \sum_{l=1}^{L} a_l(n) \cos(w_l n + \theta_l)$$

According to the researchers, this model is better suited for vocal sounds. For the purposes of our project, we can use this model with the short term Fourier transform, which shows frequency compositions over a short period of time using a spectrogram. With this tool, we would be able to understand what our source signal looks like and how to differentiate it with other background noises.

Similar to the spectrogram, researchers have proposed another method to analyze digital speech [4]. Applebaum and Hanson discuss regression features, which are defined to be linear combinations of cepstral vectors in finite time intervals. Cepstral analysis, which is the use of the inverse Fourier transform specifically on the logarithmic spectra of the signal, is used mostly in speech because it can show where the vocal excitation happens in time. In their work, Applebaum and Hanson used a recognition system based on linear prediction to analyze regression features from speech samples with varying window parameters: number of frames and time length. Although their models recognize noisy speech at relatively low rates around 50%, their work shows that prediction methods may be useful when approaching our source extraction problem.

Filtering Method

One of the most vital operations that is commonly used within the audio processing industry is the direct implementation of a specified filter within a given processing algorithm. In general, there are two types of filters that are typically implemented within audio systems: FIR and IIR filters. FIR or finite impulse response filtering provides an impulse response over a finite period, while IIR, or infinite impulse response filters, return an impulse response over an infinite duration. Both of these methods have applications within the audio industry and one may be preferred over the other depending on the exact implementation. In order to directly utilize an FIR or IIR filter within a system, typically some parameters must be known about the signal. These parameters may include type of desired filter (lowpass, highpass, bandpass, or bandstop), the attenuation of the pass and stop band, and the frequencies of the pass and stop band. In addition, FIR filters are often implemented using a window design method, which has its own set of required input parameters. Therefore, if these values are unknown, it can be difficult to develop a fully-functional audio processing algorithm using only basic filter design. However, more research is being done continuously in order to utilize filters to their maximum ability in order to process audio signals.

One particular experiment was performed by K.C. Jayashree et. Al in April of 2020 in which a team of researchers attempted to denoise audio signals in matlab using three main techniques: FIR filtering, IIR filtering, and wavelet transforms [5]. The concept of wavelets is a bit outside the scope of this paper, however, it is somewhat similar to the concept of the Fourier Transform, however, a given function is decomposed into a set of wavelets such that it can be analyzed locally in time, as opposed to globally. In order to properly compare these three methodologies, Jayashree et. Al propose an algorithm outlined by the block diagram below:

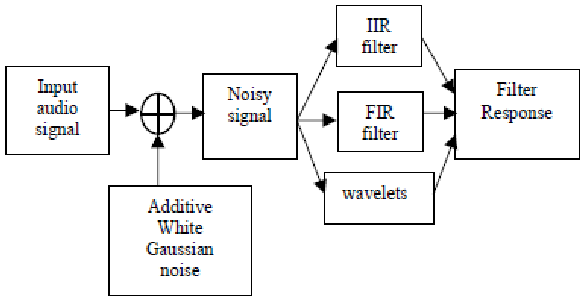


Figure 1: Block Diagram for Wavelets

In essence, an input speech audio signal was established, White Gaussian noise was added in order to cover up the speech within the signal, and then it was passed through each technique individually. For IIR implementation, Jayashree et. Al utilized results of a previous paper written by Er. Mannu Singla et. Al entitled "Paper on Frequency Based Audio Noise Reduction Using Butter Worth, Chebyshev, and Elliptical Filters" to guide their experimentation. Mannu et. Al recommended the use of a 12th order, lowpass, Butterworth filter in order to properly analyze the signal. Therefore, these inputs were implemented in the audio processing algorithm such that the results could ultimately be compared against the FIR and wavelet techniques.

For the FIR filter, Jayashree et. Al utilized a window design method to develop the fitler. For this particular experiment, due to its linear phase, a hamming window was utilized with difference equation and window function defined in Figure 2 below:

Difference Equation of FIR filter:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

Hamming Window function:

$$w(n) = 0.54 - 0.46\cos\left\{\frac{2\pi n}{N-1}\right\} \quad 0 \leq n \leq N-1$$

Figure 2: Difference Equation and Window Function for FIR Filter Design

The FIR filter was designated as a lowpass filter with passband frequency .05*π, stopband frequency, .15*π, passband attenuation of .5 dB, and stopband attenuations of 50 dB. These values were determined visually by plotting the noised audio signal in the frequency domain using the Fast Fourier Transform (FFT) and inspecting the frequencies at which speech was most likely present.

Finally, this paper analyzed the results of processing the signal using discrete Daubechies wavelet transforms. To do this, the noisy signal is decomposed into two parts: detailed coefficients and approximate coefficients. Multilevel decomposition is then performed to obtain any lower resolution components of the signal. Finally, wavelet thresholding is implemented and the signal is reconstructed. The overall process can be outlined in the block diagram below.

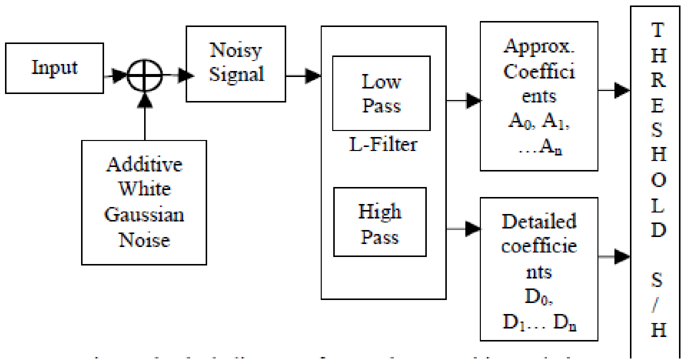

Input
Noisy Signal
Additive White Gaussian Noise
Low Pass
L-Filter
High Pass
Approx. Coefficients $A_0, A_1, \ldots A_n$
Detailed coefficients $D_0, D_1 \ldots D_n$
THRESHOLDS / H

Figure 3: Discrete Wavelet Transformation (DWT) Decomposition Technique

Overall, the results of the three techniques provide varying results. The main issue with the IIR filtering technique was that it was unstable. An analog system can be considered stable, if and only if all poles lie in the left half of the ‘s’ plane, however, this system resulted in poles lying in both the left and right halves. With this instability, the audio may get denoised a small amount, however, the output signal will be damped and distorted. Therefore, FIR filters were found to be preferred over IIR filters due to the fact that poles were only found on the unit circle and one on the origin. This provided an increase in stability, but still was not perfect. Ultimately, it was determined that wavelets were the most effective at denoising the audio, however, they are more difficult to implement than an IIR or FIR filter and require more personal tuning to implement successfully.

As a last note, Jayashree et. Al discuss the future scope of audio processing and some of the drawbacks of these filtering techniques. They determine that there are two main problems with processing audio signals using this general filtering method. First, you have to individually determine appropriate parameters for each signal you wish to process. This mainly includes desired frequencies (passband, cutoff, stopband) and attenuations (passband and stopband). In order to do this, the signal must be transformed to the frequency domain and separately analyzed before actually implementing the algorithm. Second, these filtering techniques only suppress the noise in the signal, it does not fully remove it. They suggest the future of the industry will involve actually segregating noise from speech within the signal through methods such as machine-learning and voice activity detection. As such, those practices will be discussed in detail throughout this report.

Voice Activity Detection Algorithms

Voice activity detection algorithms process an input audio signal and determine whether or not the signal contains speech. This can be used on devices to trigger a process or event, similar to how a button works, and allows humans to interact more fluidly with electronic devices. Some examples include smartphones, smart-home devices, hands-free driving in cars, and more. In these examples, it is important not only to detect speech, but also to recognize and interpret language, which allows an incredible amount of commands to be carried out by the device. Language interpretation is a highly complex problem, and is often solved using some form of machine learning. Speech detection; however, is the first crucial step in the process and this is the main focus of our discussion. There are a variety of methods currently employed for this purpose, and most work by detecting some form of change in the audio signal over time. This can be a change in the energy, frequency spectrum, or cepstral distances of the signal for example [6].

Denoising audio signals is important in a wide spectrum of applications, since signals are inherently susceptible to noise in a multitude of different formats. The applications of denoising voice signals range from public transport announcements, construction, industrial labor, military use, and anywhere else that requires clear oral communication in a noisy environment. To list some examples in the military domain, communications are important in noisy environments such as in loud engine rooms, airplanes, and ground combat, as troops must be able to coordinate with each other at all times, and verbal communication is a dynamic and effective tool to use in infinite possible scenarios. This is why the quality of voice signals is so important, and noisy environments can significantly deteriorate this signal quality if it is not filtered effectively. Therefore, denoising signals is a necessary field of study to work in concert with voice activity detection.

Methods used for voice activity detection are quite varied and abundant across a wide plethora of applications. Statistical modeling is one of the first and most widely used methods, as statistical modeling is used in most systems that require some sort of decision making capability based on a set of input conditions. The statistical method, as it is used in voice activity detection, is recapped in [7]. To formulate a model for the input or received signal, it assumes a noise signal d(t) is summed with a speech signal x(t) to obtain the noisy speech signal y(t). The short-time Fourier transform (STFT) is taken to obtain a summed signal defined as the STFT coefficient for each k frequency bin index and n frame index. Then a hypothesis analysis is performed on the binary decision of speech absence H_0 and speech presence H_1. These hypotheses are based on generated probability distributions that help make a decision about whether or not a given sample contains speech or does not contain speech. The probability distributions are assumed to be complex gaussian with some mean and variance, and a likelihood ratio test is evaluated using a combination of, *a priori* SNR, *a posteriori* SNR, and the speech spectral amplitude estimate from the previous frame obtained by a minimum mean square error (MMSE) estimator. All this to say that if the probability or likelihood ratio is above a set threshold, then the sample will be determined to contain speech, and if the ratio is below the threshold, then the sample is determined to be absent of speech. A decision is ultimately made on whether or not speech is present in the output signal based on whether or not the geometric mean of the likelihood ratios for the frequency bins is above that decided threshold, denoted as η [7].

This and similar statistical methods can be used to analyze thresholds from a variety of different methods of signal representation from frequency analysis using spectral entropy, spectral flatness, and harmonic spectral peaks to energy based analysis. One such use is an energy-based algorithm using Gaussian and Cauchy kernels as explained in [8]. This method, as is similar to how many other statistical voice activity detection algorithms function, typically extracts features from audio during periods of non-speak, and then thresholds are set based on the features found in this ambient noise sample. Once this has been accomplished, it should be relatively easy to determine once speech is present in the signal, as this threshold should be exceeded once speech is added onto the background noise. When the threshold is exceeded, a binary switch is flipped and the audio signal is classified as a speech signal. Of course, one trouble with this approach is that if the background noise changes, e.g. if a jackhammer starts going off in the distance, that the original threshold that was set is no longer accurate because aspects of the noise profile will change with the addition of this new noise to the background. This can result in a false detection of speech. Where the method in [8] differs is that the thresholds are set in a different processing space--kernel feature space--that allows the speech information to be more distinctly separated from the noise information. This is beneficial mainly because the speech and noise statistics tend to overlap when handled in the traditional euclidean space so denoising also tends to distort the desired output signal when it removes noise. It also allows the voice activity detection to work in low signal to noise ratio (SNR) environments. As such, this method transforms the input signal to a kernel feature space and executes a familiar energy feature detection to make speech determinations. The algorithm utilizes a Gaussian kernel and a Cauchy kernel, (G-KVAD and C-KVAD) and the results were compared to VAD-ZC (zero-cross rate), which is a VAD that detects the rate at which the signal crosses the value of zero. Typically, a high ZC rate means the audio is likely noise rather than human speech. These methods were also compared to a VAD-ASF (adaptive scaling factor).

An additional and more complex analysis of an audio signal is to conduct voice pattern recognition. This not only detects the speech signal as binary, but can actually classify different audio signals as spoken words. As it can be imagined, this process requires an additional level of difficulty. Dynamic Time Warping or (DTW) was a method used by Muzaffar in his paper on “DSP Implementation of Voice Recognition Using Dynamic Time Warping Algorithm” and is a technique that allows for the word of speech and the template used to identify it to be different sizes. This refers to the length or duration of the spoken word in the time domain. The size difference is a natural occurrence due to variations in word length in normal speech, and additionally causes the processing frames to be unaligned. The DTW algorithm seeks to match a reference template to each frame of the input such that the error between the reference and the input is minimized [9]. Using this method, the authors were able to achieve an identified accuracy of 70% from a provided list of 5 words, and was completely reliant on who spoke the word. In summary, statistical methods can be extremely tedious and have lackluster levels of accuracy at solving these complex audio processing problems.

The modern-day state of the art in audio processing algorithms are shifting more towards using different types of machine learning and deep neural networks. In [10], they use a boosted deep neural network which makes multiple predictions on the data at each input frame, and includes information at the input and output of neighboring frames that are pre-defined by a window size. The outputs for each frame are then averaged, and a decision is made based on the averaged thresholds. Deep neural networks is a more general term that encompasses a wide variety of network architectures. Some examples include convolutional neural networks, fully connected networks, long short term memory networks, and more. A couple of these architectures we used in our data analysis are explained in more detail later in our report. Although deep learning networks can outperform some traditional statistical model approaches, they typically do not perform as well for real-time applications which makes them more difficult to implement in practice[10].

One network architecture that does allow for real-time processing is the Convolutional Neural Network (CNN). This type of network is able to achieve a much higher frame processing rate and can therefore be used in real-time applications. In [11] a mel-scaled short time Fourier transform (STFT) using log-mel filterbank energy features is used as input to the CNNs for audio classification tasks such as speech recognition. The log-mel energy spectrum is a representation of power over a short time duration, and uses the mel-frequency scale which is described on a scale of frequencies which are subjectively judged to be equal in distance by human hearing perception. The inputs to the CNN take the form of 2D images having the mel-frequency spectrum coefficients (MFSC) on the vertical axis, and the time frames to compose the image spectrum on the horizontal axis. Filtering in this way allows the CNN kernels to find local patterns in time and frequency without considering the entire length of the signal, and yields a much higher separation in features between noise and speech characteristics. This type of feature detection is illustrated in Figure 4 below.

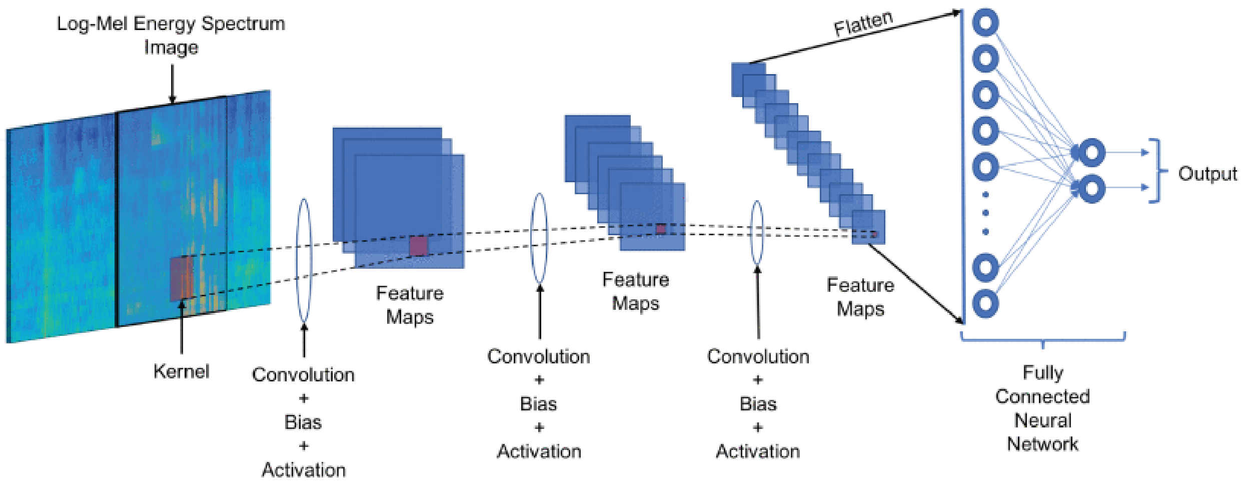

Log-Mel Energy Spectrum Image
Kernel
Convolution + Bias + Activation
Feature Maps
Convolution + Bias + Activation
Feature Maps
Convolution + Bias + Activation
Feature Maps
Flatten
Fully Connected Neural Network
Output

Figure 4: CNN usage on the log-mel energy spectrum images to detect speech in noise of an audio signal.

Active Noise Cancelling

Another application in digital signal processing is active noise cancellation (ANC). Since noise degrades the quality of audio, there were attempts to resolve this problem. The concept of noise cancellation is simple: we take the noise input and either dampen it passively with different materials or use electronics to actively cancel it. In the DSP field, the noise is received in a microphone and a new signal is generated and played against the noise to cancel it via destructive interference. In the figure below, we can see a basic control block diagram that can represent active noise cancellation.

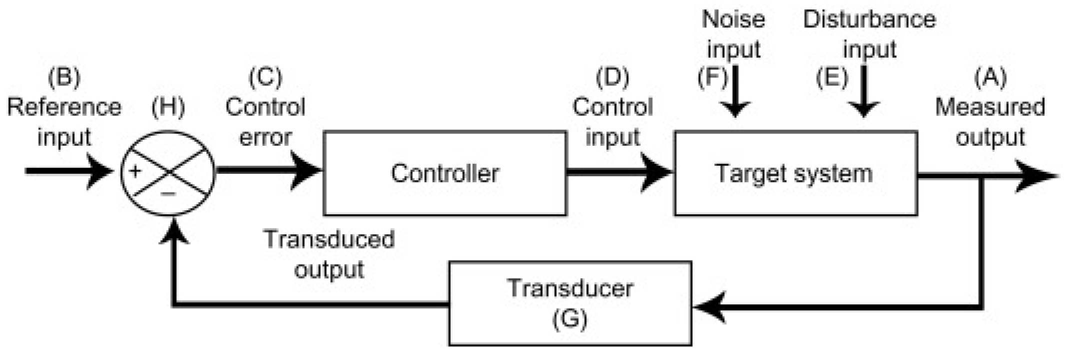


Figure 5: General Control Systems Block Diagram

Since noise is more complicated than this, researchers have created adaptive algorithms to estimate and predict noise. In several different papers researchers have proposed different applications of ANC, such for headphones, windows, and trains [12]. Although their algorithms are slightly different, their approaches are similar: taking in microphone inputs and utilizing the least-means square method.

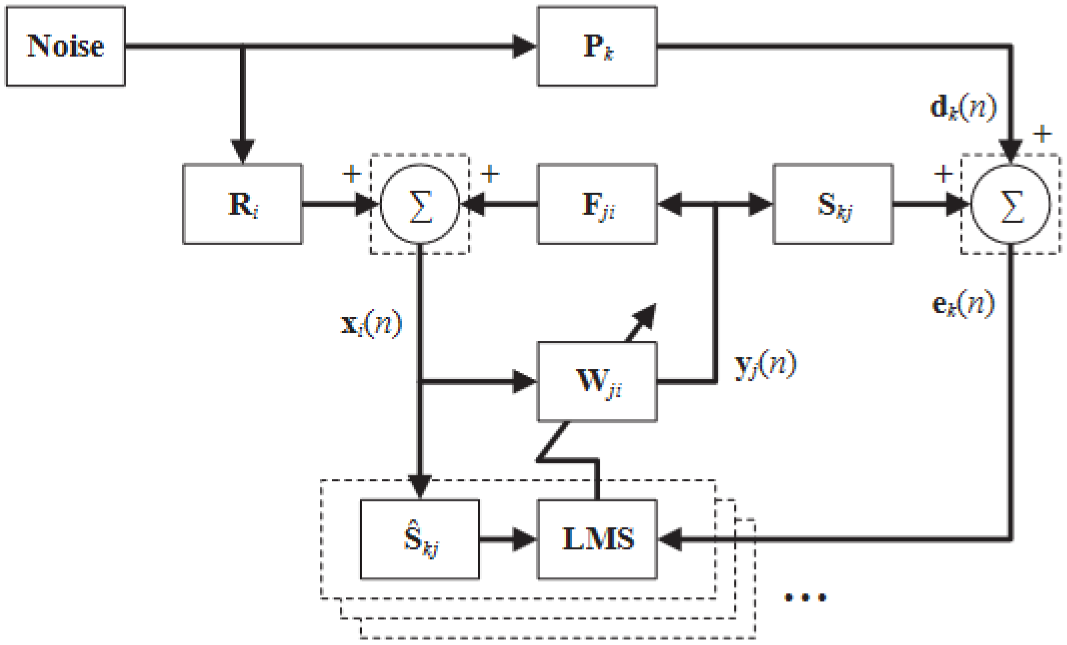


Figure 6: Block Diagram for ANC [12]

Noise would enter the microphone and go directly to the summing block via a feedforward path. A tapped delay line was then used for the to capture samples, and the output of the loud speaker would produce some noise calculated with weights and the input from the microphones.

$$y_j(n) = \sum_{i=1}^{I} \mathbf{w}_{ji}^T(n)\mathbf{x}_i(n).$$

The y(n) would then be compared to the actual noise by taking the difference and calculating the error. Since the least-means error algorithm was used, the equation above has its weights updated after every iteration.

$$J_\infty^2 = e_\infty^2 = max\left(e_k^2\right).$$

In both mentioned papers, researchers were successful in reducing noise, however there were some convergence issues, which can be fixed by changing the step size when altering the coefficients.

V. Implementation of Algorithm

Audio Data Collection

Audio samples were collected by Ethan Marcello from within his apartment in Cambridge. Background noise was created with a cheap radio emitting static noise from a poor signal through a speaker. A guitar was also utilized for background noise, specifically a Yamaha Acoustic guitar, and piano noise was generated through a pair of Mackie CR4 4" multimedia monitors playing music from Spotify. The input transducer was a Google Pixel 3XL using a free audio recording app from the Google Play Store. Spoken sentences were chosen from the "Harvard Sentences" List number 11 sentences 1 and 2, and list 62 sentence 9 [13]. These lists of sentences are known in electroacoustics to aid in testing the quality of vocal transmission. They were created from the work of the IEEE Subcommittee on Subjective Measurements in order to establish some engineering practice by which to measure speech quality. As such, they were determined by our group to be a good representation of human speech and should encompass much of the acoustic features of spoken language in english.

Voice Activity Detection in MATLAB

______As previously discussed, one method for processing audio signals is to develop a Voice Activity Detection (VAD) algorithm. Since this is such a commonly used methodology, MATLAB comes with a built-in toolbox for detecting speech using VAD. The only required input into this system is an audio file that contains human speech. In return, the algorithm will output a graph in the time domain which displays the audio signal as well as the probability of speech at every frame of that signal. The value of probability is measured at a 0 to 1 scale, returning a 1 when the system has determined that there is a 100% chance of speech being present within a given signal. In addition to this, the toolbox system object has the ability to analyze other properties of the signal, such as cepstral coefficients and pitch contour as a function of time.

The mechanisms behind how this algorithm functions are summarized in the block diagram in Figure 7:

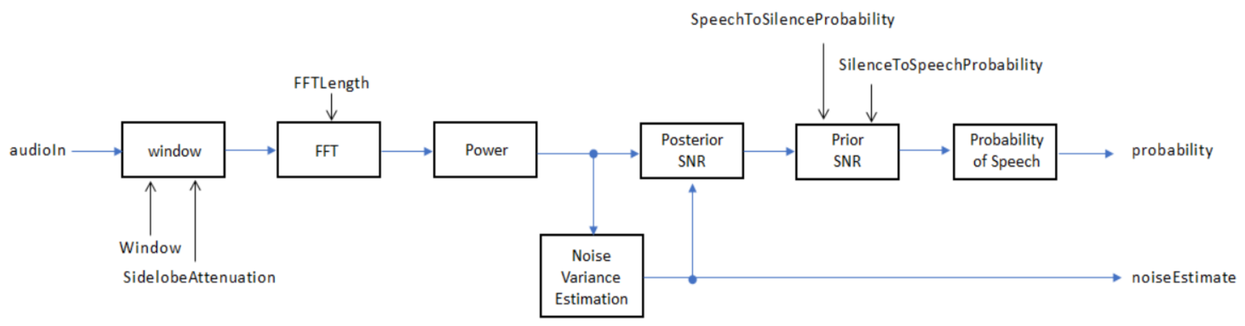


Figure 7: Block Diagram Description of VAD in MATLAB

To begin the process flow, the input audio signal must be converted to the frequency domain where it can be more aptly analyzed. To do this, the signal is windowed and the Fast Fourier Transform is utilized based on the properties of the sidelobe attenuation. The system can also accept audio inputs directly in the frequency domain, however, the system object will assume the input is a windowed discrete time Fourier Transform of a time domain signal. From the frequency domain, the power spectrum can be calculated and its prominent features extracted for analysis. Using these features, the noise variance of the signal can be calculated. MATLAB references “Noise Power Spectral Density Estimation Based on Optimal Smoothing and Minimum Statistics” by R. Martin for these calculations. Although the exact calculations are beyond the scope of this paper, Martin essentially utilizes an infinite series of power spectral densities (PSDs) over the signal, calculates their mean, and defines the noise variance based on the proportionality between the signal power and a smoothed PSD estimate [14]. Following that step, the posterior and prior signal-to-noise ratios (SNRs) are calculated using the Minimum Mean-Square Error (MMSE) formula outlined in “Speech Enhancement Using a Minimum Mean-Square Error Short-Time Spectral Amplitude Estimator” by Y. Ephraim and D. Malah. Once again, the details of these calculations are not suitable for this paper; however, the main equation utilized by Ephraim and Malah is displayed below for convenience.

$$E\{\cos\varphi_k \mid Y_k\} = \frac{\int_0^{\infty} a_k \exp\left(-\frac{a_k^2}{\lambda(k)}\right) I_1\left(2a_k\sqrt{\frac{v_k}{\lambda(k)}}\right) da_k}{\int_0^{\infty} a_k \exp\left(-\frac{a_k^2}{\lambda(k)}\right) I_0\left(2a_k\sqrt{\frac{v_k}{\lambda(k)}}\right) da_k}$$

Figure 8: MMSE Equation for SNR Calculations

where v_k is proportional to the Bessel functions of zero and first order, λ(k) represents the variances of the spectral components, and a_k and d_k are functions of the provided statistical analysis [15]. After calculating the SNRs, the algorithm assigns probability values to each frame of speech based on the model proposed by Jongseo Sohn et. Al in "A Statistical Model-Based Voice Activity Detection." This model includes a log likelihood ratio test and Hidden Markov Model (HMM)-based hang-over scheme which ultimately determines the probability value for each unit of time. Details of this advanced probability calculation can be obtained from the reference documentation [16].

In order to understand the validity of this signal processing method, our group attempted to design an experiment that utilized it within MATLAB. The exact code for this implementation can be found in the attached documentation. As previously mentioned, this method was tested using three randomly chosen sentences from the "Harvard Sentences" created by the IEEE Subcommittee on Subjective Measurements for speech quality testing. These three audio files were inputted, completely unfiltered, into the MATLAB VAD algorithm and their resulting speech probabilities were calculated utilizing the process described above. In each situation, background static was added in order to increase the noise of the signal. For each scenario, a plot was created that displayed both the audio signal in its time domain form as well as the probability counter for each frame. The resulting plots can be analyzed below in Figures 9, 10, and 11.

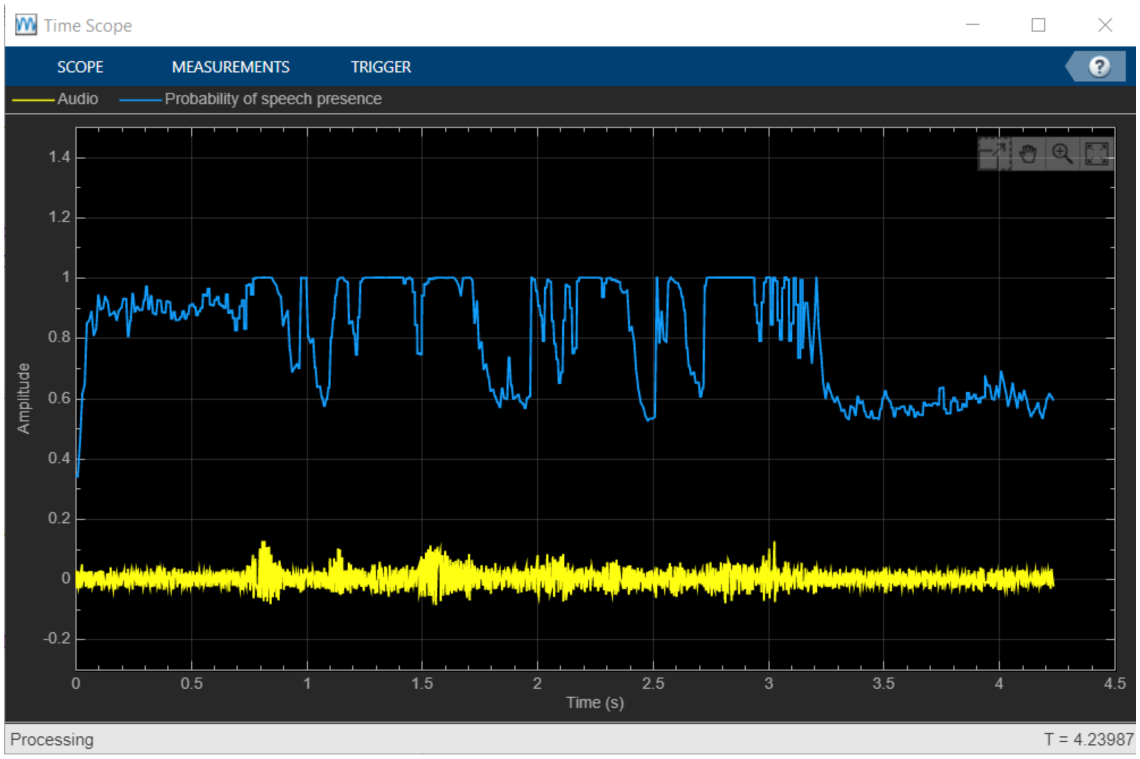


Figure 9: VAD Results of Noisy harvard_11_1 Audio File

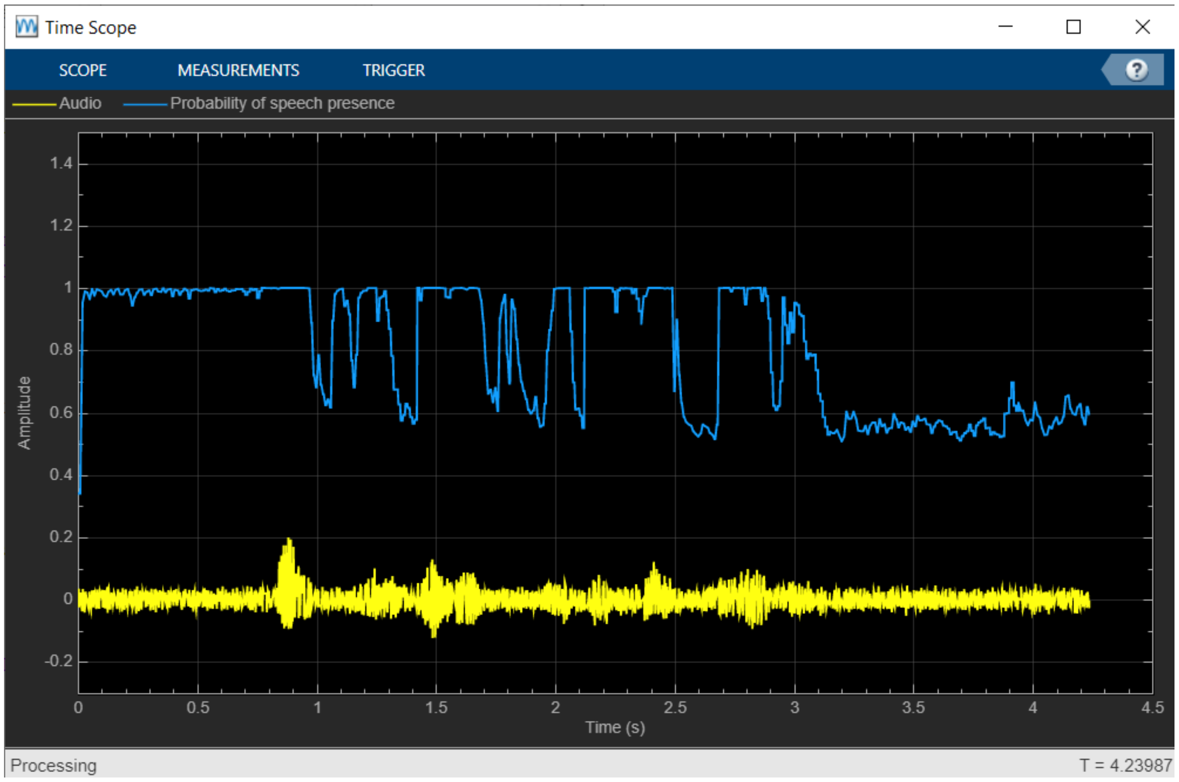


Figure 10: VAD Results of Noisy harvard_11_2 Audio File

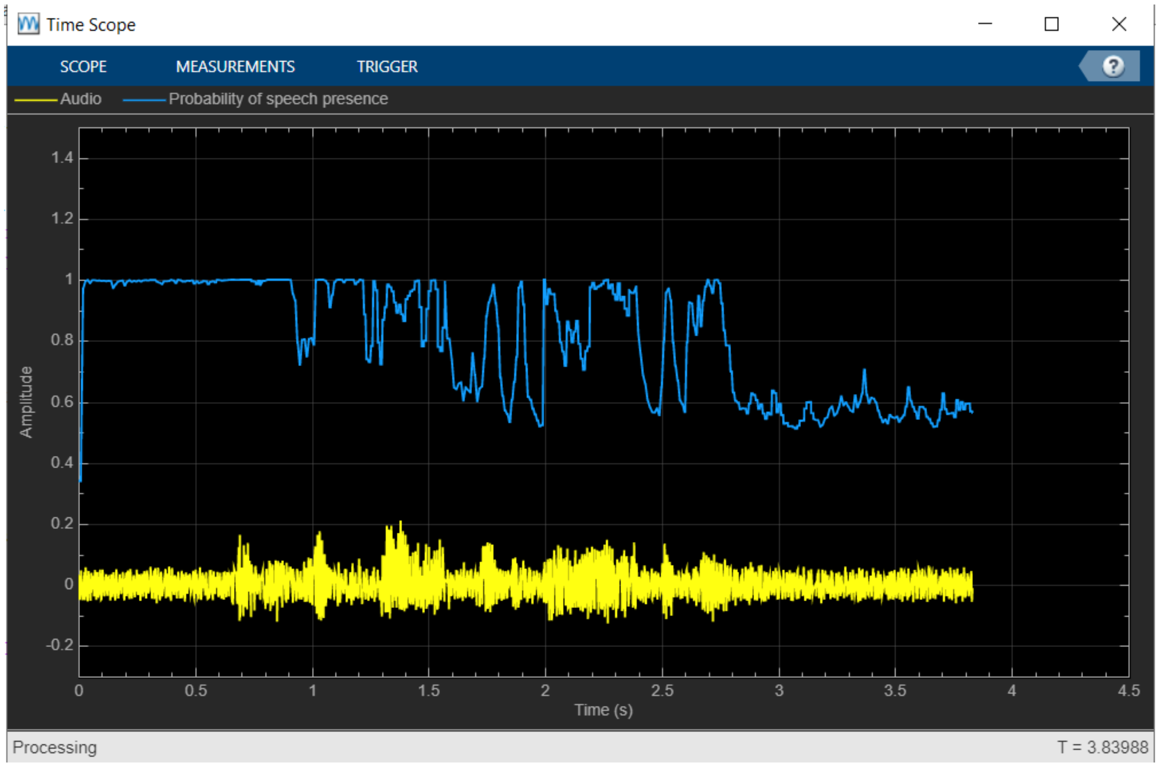


Figure 11: VAD Results of Noisy harvard_62_9 Audio File

As can be quickly be seen from observing the above plots, they are rather inconsistent in regards to whether speech is being measured.. Regardless of whether it was speech or background noise present within the original audio signal, the VAD profile was not recording accurate measurements and was incapable of distinguishing between speech and noise despite its promoted abilities. This was a surprising result for the experiment and it is not completely clear why this occurred. One possible theory is that the speech provided and the background noise were emitting frequencies that were close enough on the power spectrum to be categorized together as a high probability of speech. Similar to that the noise was likely overriding the speech in certain sections which resulted in speech not being registered in areas in which it was present. Regardless of the reasoning behind this failure, we ultimately decided that it would be inconsequential to attempt to process the signal any further given the fact that the noise was completely indistinguishable from the speech and thus no unique features could be extracted. Therefore, we continued to pursue other speech recognition methods, such as filtering and deep learning within MATLAB. Although the VAD was not suitable for this application, we determined that it could be ultimately used as a verification step to compare our noised and denoised signals. This concept will be explored later in the report.

FIR and IIR Filtering

The next processing methodology that was experimented with was filtering. As discussed earlier in the report, there are two main types of filtering commonly used to process audio signals: IIR filters and FIR filters. Filters were constructed in MATLAB using both of these methodologies and respective experiments were performed on three different audio files: harvard_11_1, harvard_11_2, and harvard_62_9 as previously discussed. For the sake of simplicity, any figures contained within this section will refer only to audio processed from the harvard_11_1 file. Information on the other sources will be summarized here and presented in full in the attached documentation.

To begin the experimentation, general filter parameters had to be established for utilizing desired values in the frequency domain. In order to convert the time domain signal to the frequency domain, the Fast Fourier Transform was utilized within MATLAB. Figure 12 below details the time domain and frequency domain parameters of the signal before any processing occurred.

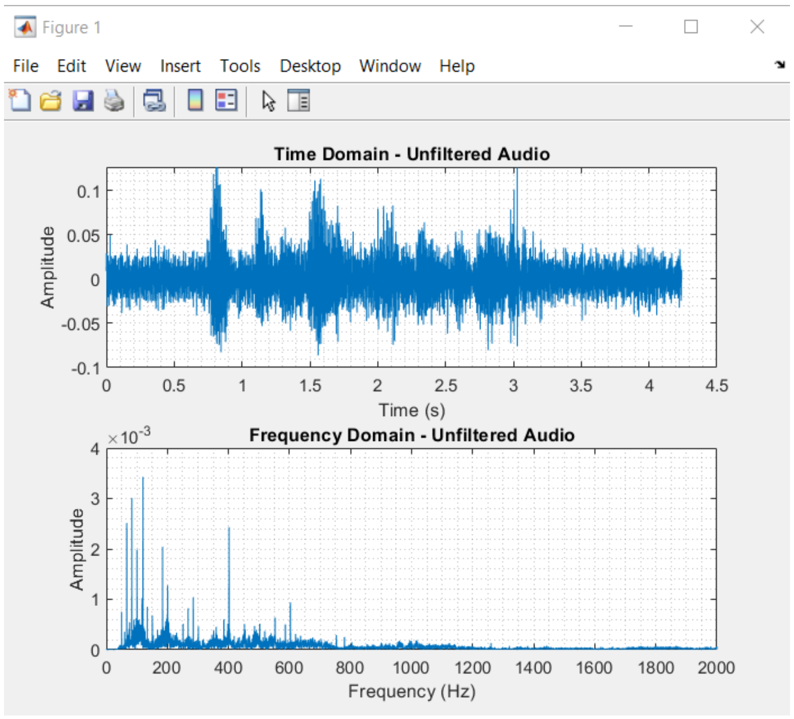


Figure 12: Time and Frequency Domain Representation of harvard_11_1

From the frequency domain description of the signal, it is necessary to extract the features that will be utilized in the filter analysis. According to the National Center for Voice and Speech, the fundamental frequencies of human speech generally occur within the range of 0 - 300 Hz [17]. Therefore, the filter parameters should be decided such that values in that range are allowed to pass through, but the values outside are attenuated. In order to determine the filter attenuations, we referenced the work Jayashree et. Al and the values at which they selected. In addition, since the frequencies we wish to pass are limited to the low end of the spectrum, it was determined that a lowpass filter would be acceptable for both IIR and FIR applications. Table 1 below summarizes all the filter parameters that were ultimately determined based on this criteria. Some of the values, such as sampling frequency, were automatically calculated by MATLAB as it reads the audio file; these were not selected by the group. The normalized frequencies were obtained by dividing the original values by the given sampling frequency.

Table 1: General Filter Parameters

Parameter Name	Parameter Value
Passband Frequency (Fp)	300 Hz
Stopband Frequency (Fs)	1000 Hz
Sampling Frequency (fs)	16000 Hz
Normalized Passband Frequency	.01875
Normalized Stopband Frequency	.0625
Passband Attenuation (Ap)	0.5 dB
Stopband Attenuation (As)	50 dB
Normalized Cutoff Frequency	.0406

After establishing the filter parameters, two separate algorithms, FIR and IIR, were developed. Beginning with the FIR implementation, we decided to base our model on the design specified by Jayashree et. Al which involved the windowing method. For this particular application, a Hamming Window was used with a length described below:

$$L = \frac{6.6*\pi}{\Delta w}$$

where Δwcan be specified as the difference between the passband and stopband frequencies. Using the parameters from Table 1, the resulting length of the window is 151. The full code implementing this FIR filter can be found in the attached documentation; however, its windowing function and magnitude response has been included in Figures 13 and 14 below for verification of functionality at the prescribed parameters.

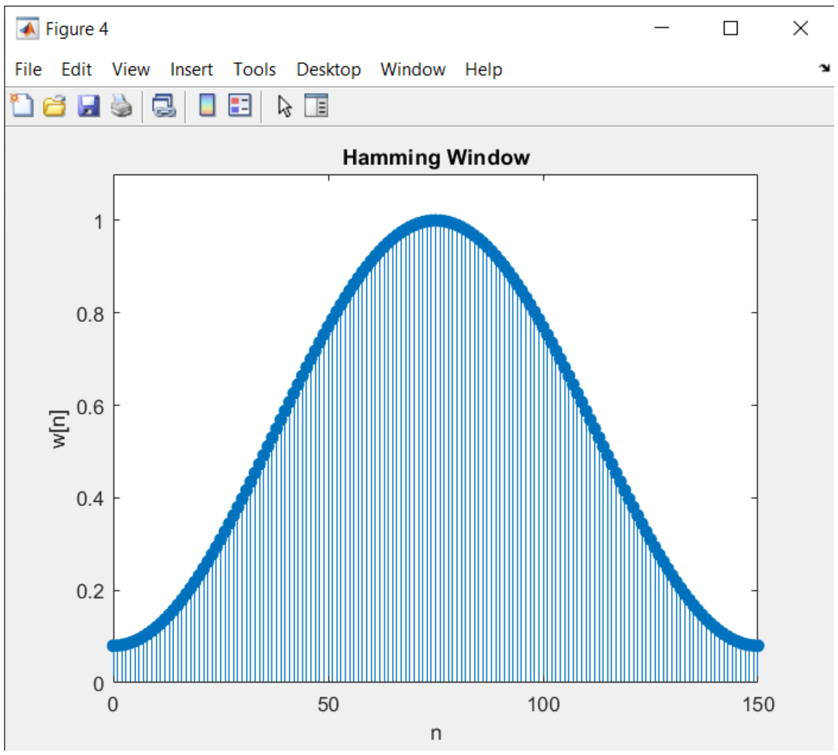


Figure 13: Hamming Window Function

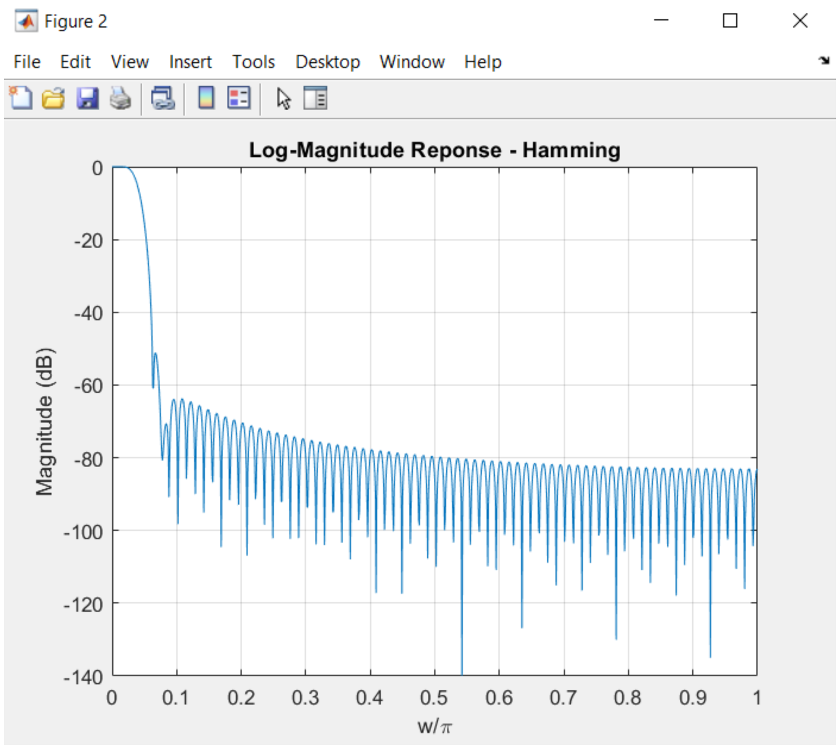


Figure 14: Log-Magnitude Response of FIR Filter with Hamming Window

As can be seen, the filter displays passband, stopband, and cutoff frequencies at the specified locations. In addition, there is a minimum 50 dB attenuation present at all values past the stopband frequency. Utilizing this filter, the original audio signal was processed and the excess noise was removed within the limitations of the filter. After implementing the filter, the resulting signal was represented in the time domain and compared with the original audio file. The resulting plots can be found in Figure 15.

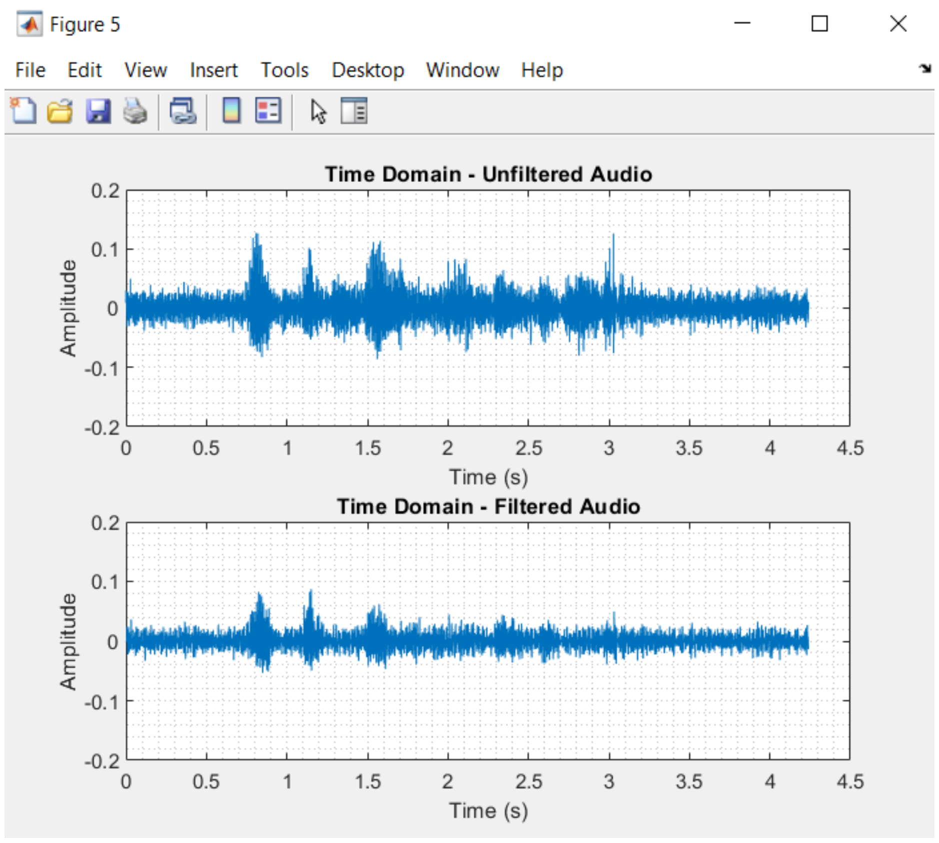


Figure 15: Time Domain Representation of Original and FIR Filtered Audio Signal harvard_11_1

As can be seen, the filtered audio signal contains much less noise than the original. The main signal features that remain present are those which relate to the points in time in which speech is clearly present. Visually speaking, it is clear that this filter was successful at reducing the amount of noise within the signal. However, when played pack as an audio file, it was clear that there was some distortion present within the filtered signal. While there was hardly any audible noise, the speech sounds also appeared to be muffled and suppressed. This same pattern was repeated across all three audio files tested, meaning it is unlikely this was a file specific issue. It is possible that, while focused on reducing noise, the filter also removed some frequencies of speech that were vital to composing a fully understandable signal. This was a clear downside to this methodology.

After implementing the FIR filter, an IIR filter was also designed such that the two could be compared. Modeling after Jayashree et. Al, a butterworth filter was chosen for implementation within this system. The order and cutoff frequency of the filter was determined by using the MATLAB command "buttord" with system inputs described by Table 1. The system coefficients were then calculated using command "butter" such that the filter could be implemented on the input audio file. The log-magnitude response of the filter can be seen in Figure 16. The time domain representations of the unfiltered and filtered signals can be seen in Figure 17.

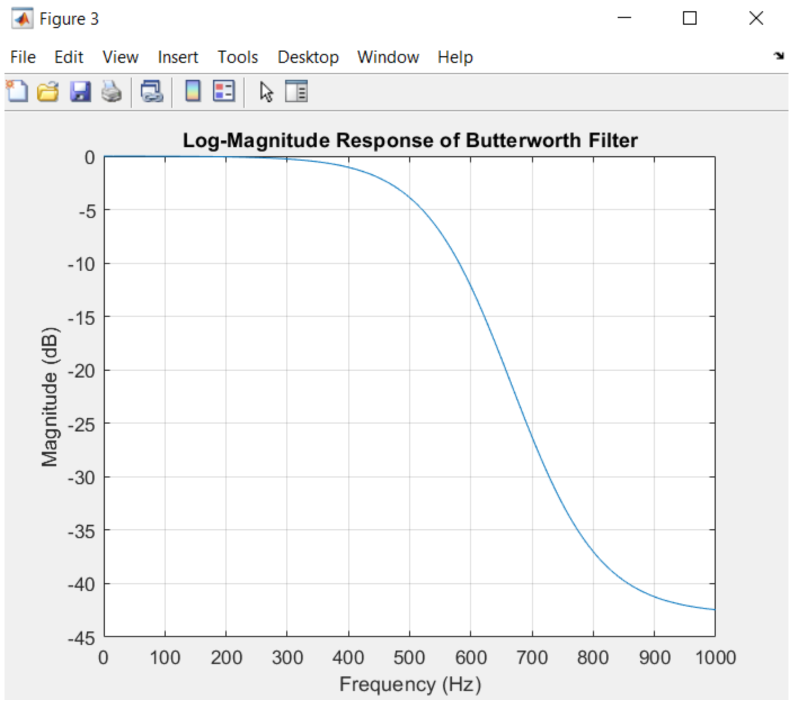


Figure 16: Log-Magnitude Response of IIR Butterworth Filter

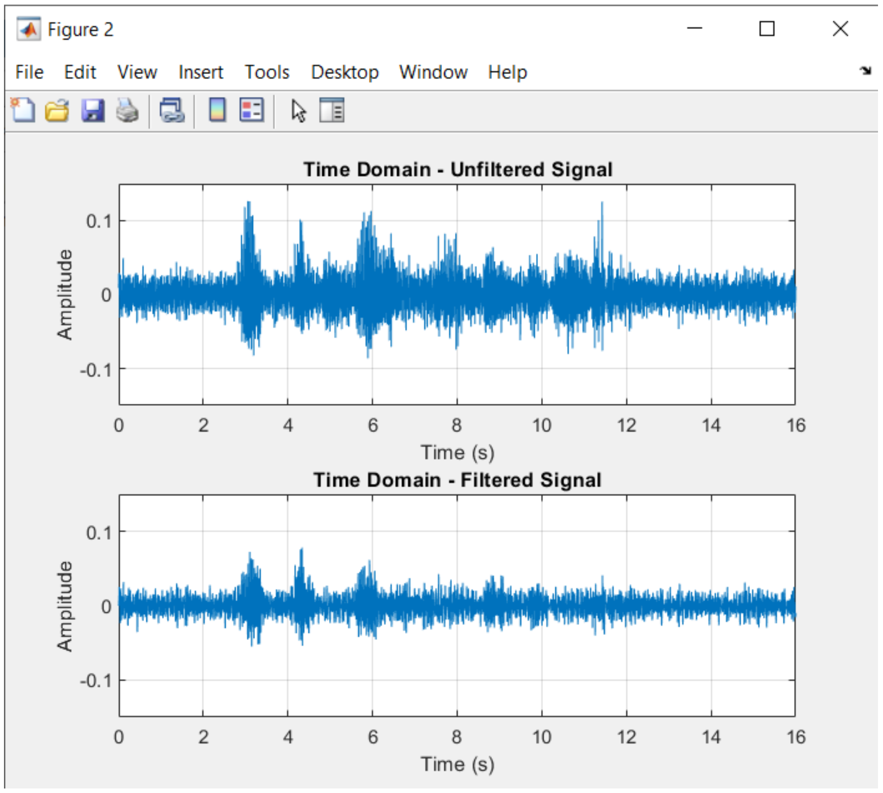


Figure 17: Time Domain Representation of Original and IIR Filtered Audio Signal harvard_11_1

As can be seen, the filter clearly removed much of the noise present around the peaks of the audio signal. However, we again found that the speech sound was muffled when played back out loud. This pattern repeated over the course of all three audio files tested, similar to the circumstances surrounding the FIR filter. Based on this fact, it was assumed that some of the important frequencies for speech recognition were being suppressed by the parameters of both filters. This was ultimately determined to be the main drawback of this method, since it is difficult to suppress background noise without also suppressing speech. This was a concern also mentioned by Jayashree et. Al during their experiment. Since unwanted audio is being suppressed, but not fully segregated, it is difficult to utilize this method to fully process a given signal without some distortion becoming apparent.

Denoising Speech Using MATLAB Deep Learning Networks

Since modern-day state of the art voice recognition techniques are frequently using neural networks and machine learning to achieve increased levels of performance, we decided to find an implementation in MATLAB that we could replicate and test on our own data, comparing it to the results obtained from basic digital signal processing filtering techniques. The tool we found was the "Denoising Speech Using Deep Learning Networks" article, which enlists the use of MATLAB Audio Toolbox™, Deep Learning Toolbox™, and Signal Processing Toolbox™ to take in an audio signal, add noise, utilize a deep learning network to identify features in the audio signal attributed to human speech, and denoise the noisy signal based on these feature detections [18].

One method that was used to filter out noise was with deep learning neural networks. These networks are used to find discriminating features from an input signal to detect a speech signal and remove the noise. A basic diagram of a neural network is shown below. It is composed of nodes (circles) and unit connections.

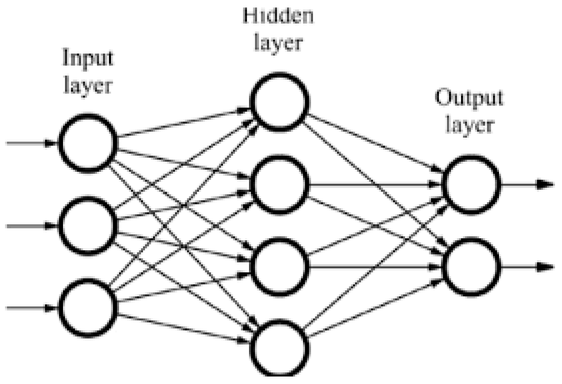


Figure 18: Basic Neural Network

Each of these connections has a certain value (or weight) assigned to it, depending on the design parameters. The nodes have a certain function and threshold for it to "activate". The combination of connections and nodes form what are called layers. In the mentioned figure, the input layer is where the data is sent into the network, the hidden layer is where the data is processed, and the output layer produces a result. When a network has multiple hidden layers, it is considered a deep neural network. These can also be enhanced to "learn" by adjusting weights after each input by forwards and backwards propagation, which are used to minimize the error of some cost function. Additionally, layers can be fully connected, meaning that all inputs go into each node in the next later, or convolutional, which only makes certain connections to a given neuron.

Because of this, convolutional layers are considered more specialized and can be more efficient, while fully connected layers are for general use.

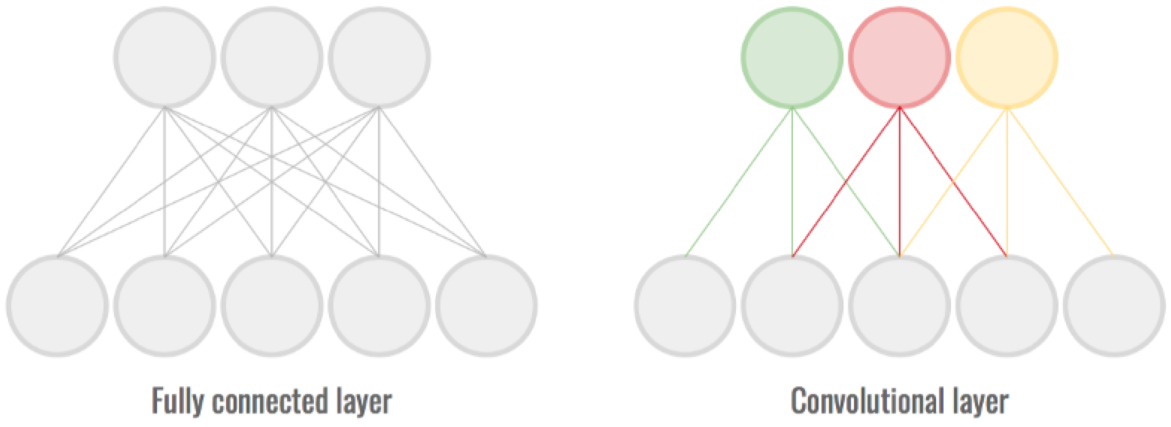


Figure 19: Visual for Different Connected Layers

In the MATLAB Deep Learning Toolbox™, we utilized both a pre-trained neural network, as well as our own trained network in order to denoise a noisy signal. Both the trained network and our own trained network were trained on the same training dataset. The training set was based on a public database called Common Voice (by Mozilla), which features many samples of human speech. Below in Figure 20 is a block diagram describing the functionality of the network as a tool to denoise audio signals. As we can see, a noisy signal enters the network after having the Short Time Fourier Transform applied to the time-domain signal, and its output is the predicted magnitude of the original audio signal, which is combined with the angle of the original noisy audio signal in order to create a denoised signal.

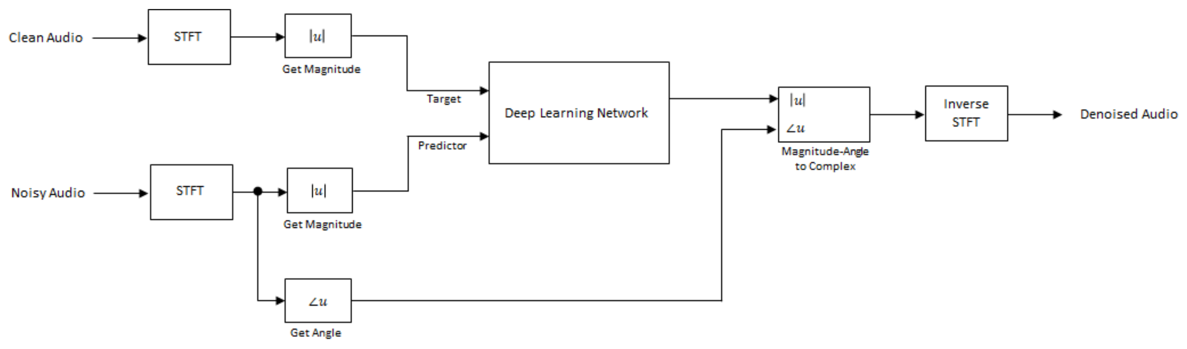


Figure 20: Block Diagram showing the denoising tool functionality.

The network can be configured to be fully connected or convolutional, with each having their own advantages and disadvantages. Both methods were used in this paper to determine whether there were significant differences between them.

The Short-Time Fourier Transform (STFT) is used to convert the time domain audio signal to the frequency domain to be analyzed by the deep learning network. The STFT implements a hamming window of length 256, and is periodically applied over the data with an overlap of 75% of the data samples. This generates STFT vectors across the data to be used in the deep learning network. During the training stage, 8 consecutive STFT vectors from the noisy audio are passed into the deep learning network as the predictor data, which allows the output estimate to be computed based on both current and past data. A single STFT vector is used as the target for the learning network. Tall arrays are then used as an efficient way to extract features from the STFT data, as the calculations are queued until a certain function is called. This way the features can be extracted by running calculations in parallel which ensures the minimum number of passes through the data. This feature requires the use of the Parallel Computing Toolbox™.

During the network training phase, the network receives both the predictor and the target STFT vectors--the noisy audio and the clean audio respectively--and outputs an estimate of the target vector. This estimate is then compared to the original clean audio vector, and the RMSE (root mean square error) and loss is calculated, which are each measurements of how well the network is achieving the desired target output--i.e. the clean audio. The RMSE is a calculation of how far the individual data points are from each other, and loss is a calculated cost function that the network is trying to minimize. The weights of the connections between the nodes are adjusted appropriately to minimize this function. From the training data, one percent of the total dataset is used for validation, which ensures that the network is denoising the audio signals correctly. During the validation step, the network doesn't receive the clean audio file at the input. The network takes the noisy audio input and produces an estimate of what the clean audio should be as an output. If the RMSE and loss corresponds to the same values as in the training, then the data is being fit appropriately. An example of the training data plot for the convolutional layers network is included in the appendix.

The MATLAB provided denoising algorithm makes a few assumptions in its denoising algorithm, which makes it more effective at the given example application than for other applications. It is entirely possible for this algorithm to break down if these assumptions aren't maintained. The first assumption that the algorithm makes is that the noise source is a random sample from a provided audio file "washing machine noise". This noise type has specific characteristics that make it easier to detect and filter out. A sample of the noise and some relevant statistics are shown in Figure 21 below:

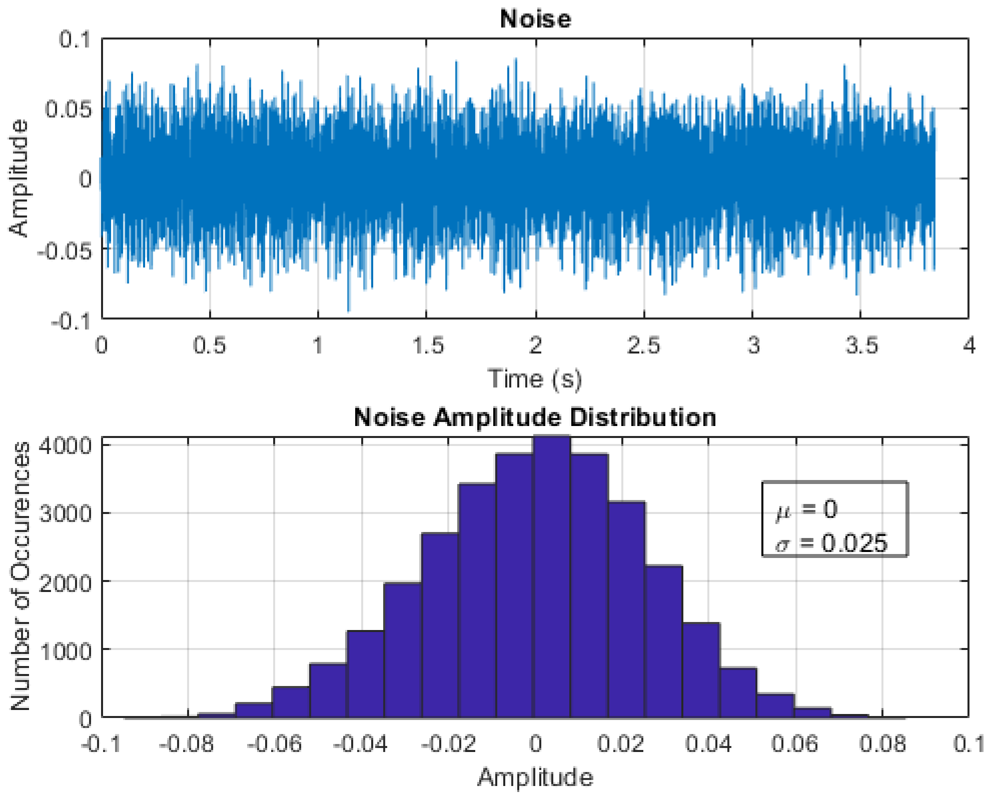


Figure 21: Plots of the noise signal and the noise distribution of a random sample of the "WashingMachine-16-8-mono-1000secs.mp3" MATLAB file.

As can be seen from the plots, the distribution of the noise used to train the network predictors was approximately Gaussian with zero mean. This type of noise is well-studied, and has a wide variety of applications as it is naturally occuring in nature. Therefore it is a fair assumption to make that the noise is Gaussian; however, this means that the network may not be as effective at reducing noise that is not Gaussian.

To test this theory, we will use in our analysis a noise source that has a different statistical profile than the washing machine noise. An audio sample from piano music was downloaded and sampled from a free audio website and a random sample was used to add over the clean audio signal for analysis by the networks. The plots of the time domain sample signal as well as its amplitude distribution are shown in Figure 22.

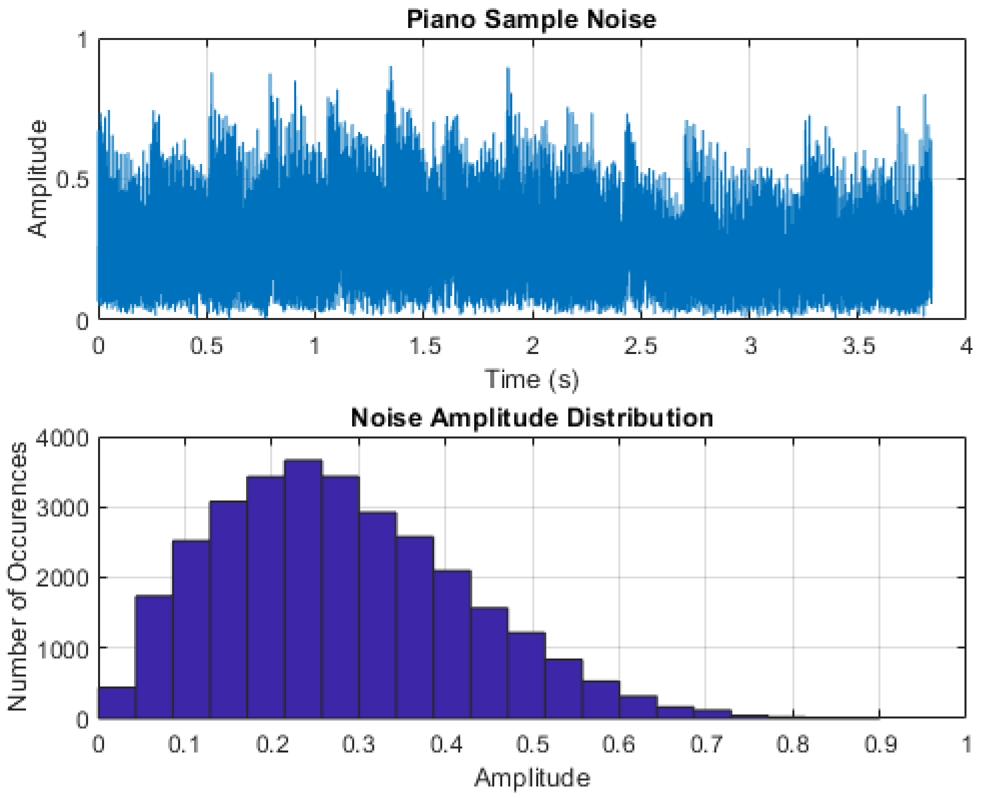


Figure 22: Piano noise sample plotted in the time domain as well as its distribution. Recording was taken from “relaxing_piano.mp3”

It is easy to see by the distribution that the piano audio distribution is non-Gaussian, and additionally has a mean value greater than zero. As such it should be well suited to test the robustness of the network output by using this noise profile.

The second assumption made in the MATLAB algorithm is that the signal to noise ratio (SNR) between the speech audio and the noise is zero decibels--i.e. the signal powers are equivalent. This assumption is likely enforced to prevent the noise from drowning out the speech signal to make it undetectable. If the noise power is greater than the speech signal power, then it becomes increasingly more difficult to detect the signal and filter out the noise. As such, it is worth noting that if we allow the SNR to be less than zero decibels, then this network should not be effective at denoising the speech signal.

The level of success achieved using the deep network algorithms was substantially greater than the results from our low-pass filter approach. Denoised audio was much less distorted, and noise was sufficiently removed from the original signal. A comparative analysis is done in this section between the performance of the fully connected network and the fully convolutional network, and additionally mentions their performance relative to the FIR and IIR filters.

First, a qualitative determination was made by observing the audible difference between the noisy audio file and the denoised audio file for both the fully connected and convolutional networks, and was completed for both the standard noise profile as well as an unexpected noise variant--piano music. Overall, when listening to the noisy speech signal, we found that we were still able to generally hear and understand the speech over the noise by ear. This indicates that from a qualitative standpoint, the added noise that is filtered out is not enough to severely distort the input signal. Additionally, the denoising may have eliminated some of the noise, but it is important to recognize that this comes at the cost of a possibly degraded speech signal. Therefore, depending on the application, denoising may not be the best approach to recovering desired information from a signal, since the speech was qualitatively not much more intelligible between the noisy and the denoised audio. In every case, the perceived noise was undoubtedly reduced for both networks in both types of noise. The convolutional network seemed to do a better job at muting additional noise, but this also caused the denoised speech signal to sound somewhat distorted. This was especially pronounced when the piano noise was used, as the piano was almost impossible to hear after the noisy audio was denoised by the convolutional network. This was a rather surprising result since we didn’t anticipate the network to work as well when a different noise source type was used.

Moreover, a more quantitative comparison was conducted on the performance of the deep learning networks handling both types of noise. Comparisons were done in both the time domain as well as the frequency domain, and the root mean square error of each signal pair was assessed. The time domain analysis is seen for one speech sample in Figures 23 and 24.

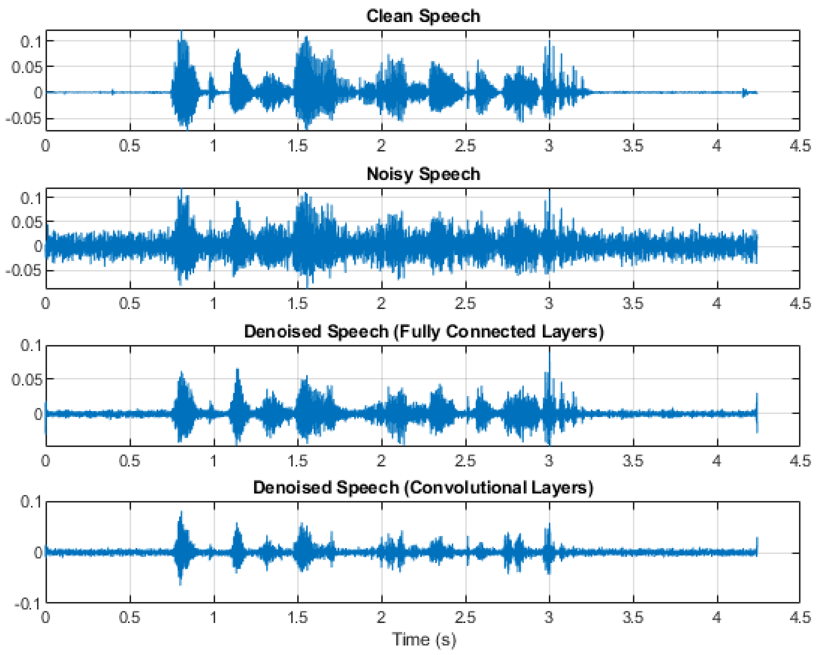


Figure 23: Time domain plots of the harvard sentence list 11 number 1 with washing machine noise for clean, noisy, and denoised using both the fully connected and convolutional network.

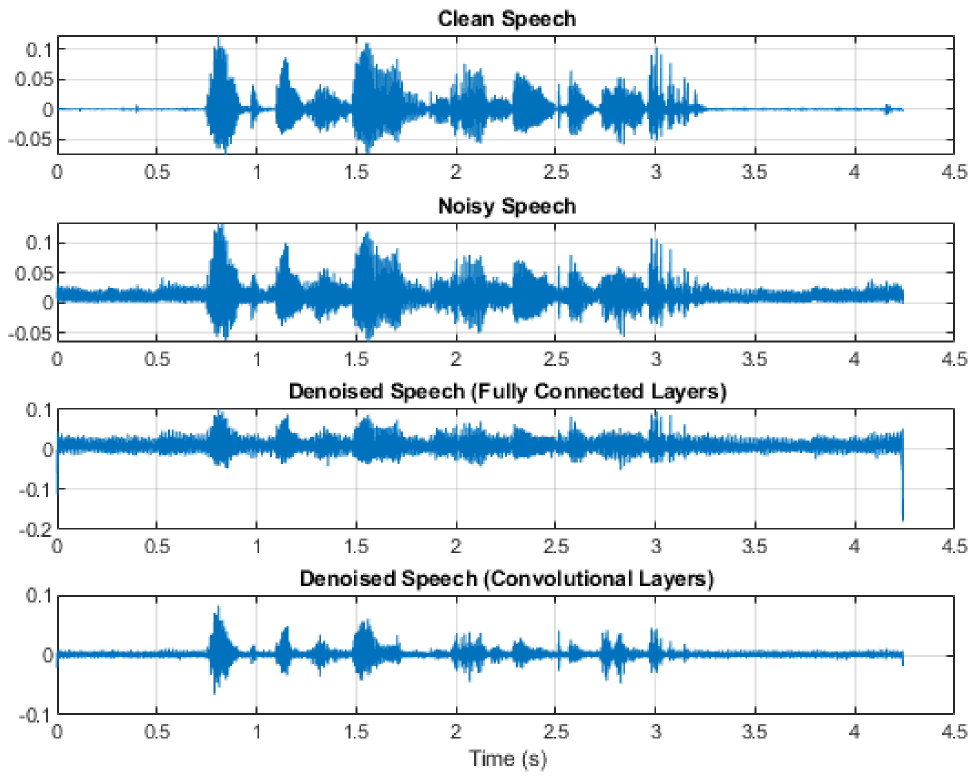


Figure 24: Time domain plots of the harvard sentence list 11 number 1 with piano noise for clean, noisy, and denoised using both the fully connected and convolutional network.

As is seen from the plots, the deep learning networks removed most of the noise components in the noisy signal, however, there are some features that were removed from the original signal as well. This is more evident in the convolutional layer, where important features at each excitation were noticeably altered by the network. When these signals were played back in MATLAB, the original speech was able to be heard in the denoised speech, but the quality is not the same. One thing to note is that when the piano noise was applied, the neural network seems to have degraded performance. This is due to the network not being familiar with this noise profile, since it was only trained using the washing machine noise on top of the human speech. Additionally, for the fully connected layers denoising the piano noise, it does not seem to remove the noise as well which could be due to similarities in features between the piano vs. speech. For the convolutional layer; however, it seemed to do a better job at removing noise.

Additional analysis was performed in the frequency domain in order to determine the effectiveness of the deep learning networks. As in the time domain, each signal from the clean audio, the noisy audio, and the denoised audio was included which were analyzed in two separate figures that contain either the washing machine noise or the piano noise. The spectrograms can be seen in Figures 25 and 26 below, where they correspond to the washing machine noise and piano noise respectively.

Taking a look at the spectrograms, it is easy to see that the noise covered a broad range of frequencies, as the Noisy Speech spectrogram is more yellow and green in all areas of the image than the original clean audio spectrogram. This is especially apparent in the lower frequency range. As can be seen, the denoised speech files removed a lot of the extra power increases, and many of the defining shapes of the clean spectrogram can be better seen in the denoised version(s) than in the noisy spectrogram. What is apparent right away is that the denoised spectrogram using the fully connected layers network for the piano noise is covered in high power signals that are scattered across both the time and frequency axes. Many of the power increases are within small frequency bands at seemingly regular intervals across the spectrum, which is quite curious. To the eye, this is the only example from the denoised samples that produced a spectrogram that looked more cluttered and noisy than the noisy spectrogram.

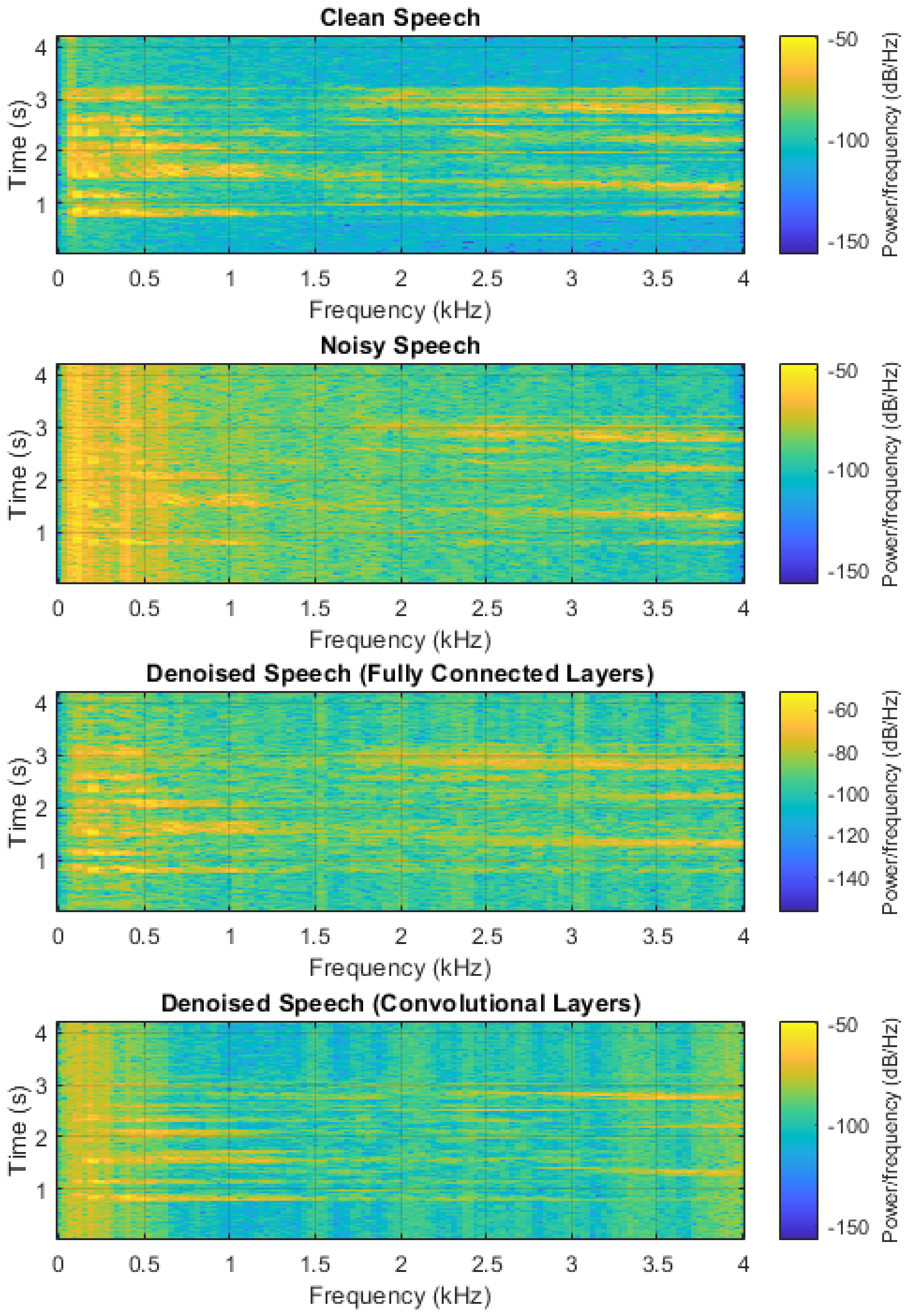


Figure 25: Spectrograms of clean, noisy, and denoised signals for the harvard_11_1 dataset with washing machine noise.

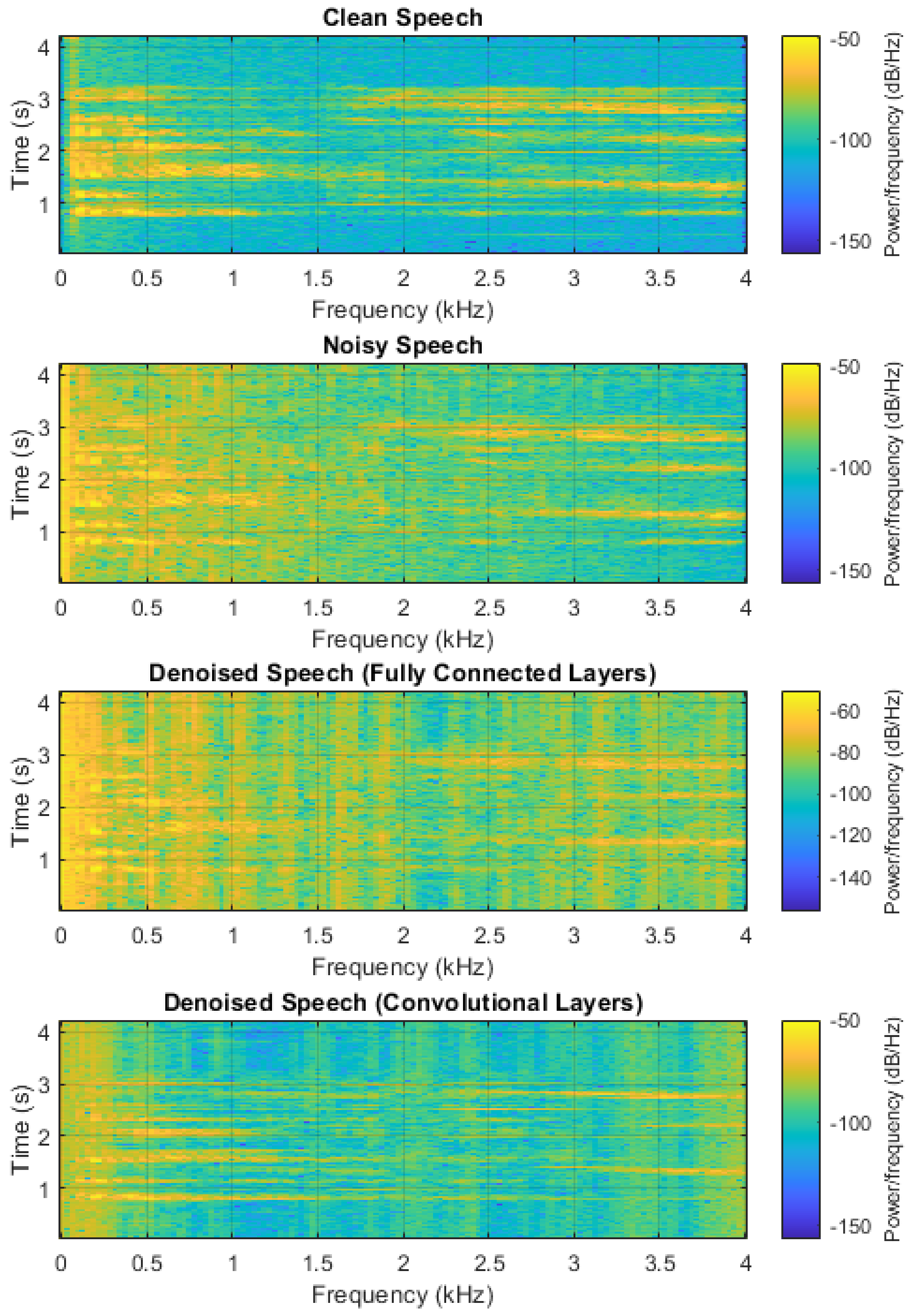


Figure 26: Spectrograms of clean, noisy, and denoised signals for the harvard_11_1 with piano music noise.

These plots alone support that the deep networks were effective at removing noise, but they do not display a quantitative measurement to determine their relative performance. As such, the RMSE was calculated in table 2 and table 3 below for the noisy and denoised signals in comparison to the clean audio signal to get a better idea of their relative performances. The RMSE was calculated below:

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y_i} - y_i)^2}{n}}$$

Where $\widehat{y}_i$ is the estimated signal, y_i is the clean audio signal, and n is the total number of samples. Table 2 contains the RMSE data for the FIR and IIR filters, while table 3 contains the RMSE data for the fully connected and fully convolutional denoising networks.

Table 2: Comparison of RMSE data between the noisy audio and the denoised audio using FIR and IIR filters for washing machine (wm) noise.

Dataset	Range	Noisy RMSE	FIR RMSE	IIR RMSE
harvard_11_1 with wm noise	0.2127	0.0126	0.019	0.0189
harvard_11_2 with wm noise	0.3237	0.0139	0.0203	0.0199
harvard_62_9 with wm noise	0.3376	0.0213	0.032	0.0311

The “Range” criteria is the minimum value noisy audio sample subtracted from the maximum value noisy audio sample. This provides a metric by which to assess the relative RMSE performance with respect to the maximum value seen in the signal. Standard deviations of the residuals were considered as an additional comparison metric, but they were left out of the chart as they were equivalent to the RMSE values in each case. “FIR” represents the finite impulse response filter which used the windowing method with a hamming window, and “IIR” represents the infinite impulse response using the butterworth filter. From this table, we can see that the RMSE values seem to be higher when we apply the filter. What this means is that the resulting filtered signal is actually "worse" for the FIR and IIR filters.

Table 3: Comparison of RMSE data between the noisy audio and the denoised audio with the deep learning networks for washing machine (wm) and piano noise.

Dataset	Range	Noisy RMSE	FC RMSE	CL RMSE
harvard_11_1 with wm noise	0.2235	0.0126	0.0080	0.0091
harvard_11_2 with wm noise	0.3461	0.0139	0.0088	0.0094
harvard_62_9 with wm noise	0.3125	0.0213	0.0129	0.0140
harvard_11_1 with piano noise	0.2004	0.0126	0.0136	0.0087
harvard_11_2 with piano noise	0.3359	0.0139	0.0151	0.0089
harvard_62_9 with piano noise	0.2844	0.0213	0.0224	0.0129

Alike table 2, the "Range" criteria is the minimum value noisy audio sample subtracted from the maximum value noisy audio sample, and the standard deviations of the residuals were left out of the chart as they were equivalent to the RMSE values in each case. "FC" represents the fully connected network, and "CL" represents the fully convolutional network denoised signals on the respective dataset. In general, the networks outperformed both the IIR and FIR filters, but the fully connected network does show worse performance when the piano noise is introduced to the signal, supporting our observation of the scattered frequency power in the spectrogram analysis. Ignoring this exception, we see that the convolutional layers generally performed well in removing noise in terms of the RMSE.

From the table 3, it seems that there is a direct correlation between increases in the noisy

RMSE value and the denoised RMSE value for both networks. Although there were not enough data to prove this to be causation, the result is expected. As the noisy signal becomes increasingly more different from the clean signal it should be more difficult to accurately recover the original signal. This is true for both networks in the presence of each noise type. The noise type did however play a factor in determining the relative performance between the two networks. For the washing machine noise--the expected noise--it seems that the fully connected network was more accurate at recovering the clean audio signal than the convolutional network. This is shown by the fact that the RMSE values were slightly lower using the fully connected network for each dataset with this noise profile. On the other hand, when an unexpected noise profile is used like the piano music--ie. a noise profile not used in training, then the result was the opposite. In this case, when the network was used on data with untrained noise characteristics, the convolutional network outperformed the fully connected network. This is an interesting result, and would indicate that in the presence of varying/unpredictable noise variants, a convolutional network may be a more attractive option.

In either case, both networks did obtain an increased level of performance across the tested data (with the exception of the fully connected network in the presence of piano noise), and although one network may be slightly better for different applications, the difference may not be enough to warrant the use of one over the other when other factors are not at play. For many real time applications; however, the convolutional network may be the preferred method due to its faster processing speed.

The deep learning networks approach to denoising an audio signal is in fact a highly effective method for reducing unwanted noise while retaining as much of the original speech quality as possible. It is reasonable to believe that even better performance can be achieved if a larger training dataset is used or a different representation of the audio signal is passed into the network that can allow it to extract better features to help the network learn from the data. We were able to achieve satisfactory performance with the suggested MATLAB tools by running our own data, so we didn't believe any sort of enhancement was necessary, but it may be worth exploring in future work. Additionally, training the network with varying types of noise profiles could increase the robustness of the denoising prediction for audio files containing non-Gaussian noise, especially if used in conjunction with a source separation algorithm.

VI. Comparison and Verification of Results

Now that different audio processing methods have been analyzed, it is important to compare the results to fully understand their individual benefits and drawbacks. Due to the inability of the MATLAB voice activity detection (VAD) algorithm to properly differentiate between speech and noise, it will be excluded from this comparison. However, the VAD can be used in order to verify the overall success of the other two methods. If the signals have been denoised properly, the VAD tool in MATLAB should produce proper speech recognition plots that are not compromised by excess noise.

Before analyzing the signals using the VAD, it can be useful to simply examine the frequency representation of each processed audio signal compared to the original. Figures 27 and 28 below detail this concept.

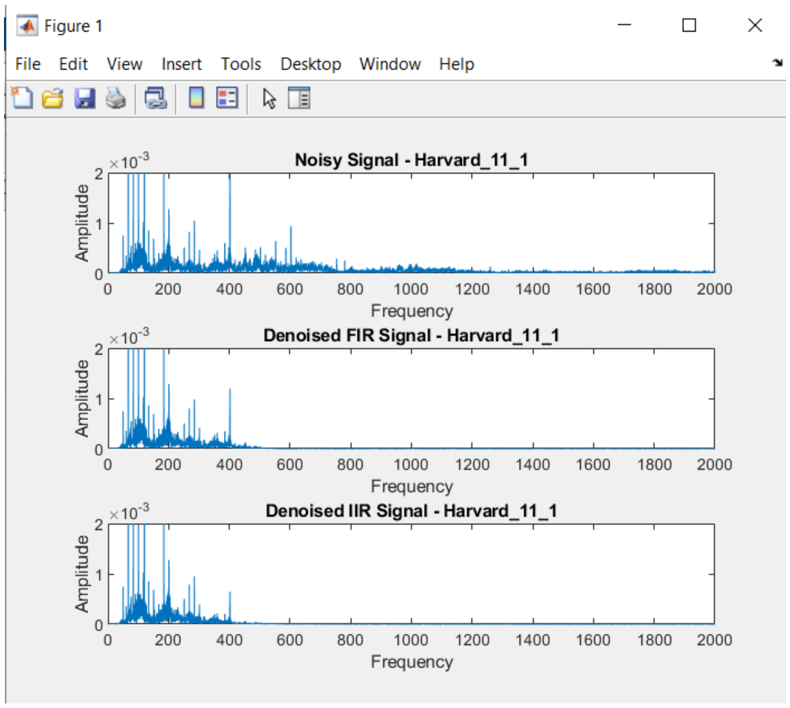


Figure 27: Frequency Representation of Original Signal and Filtered Signals

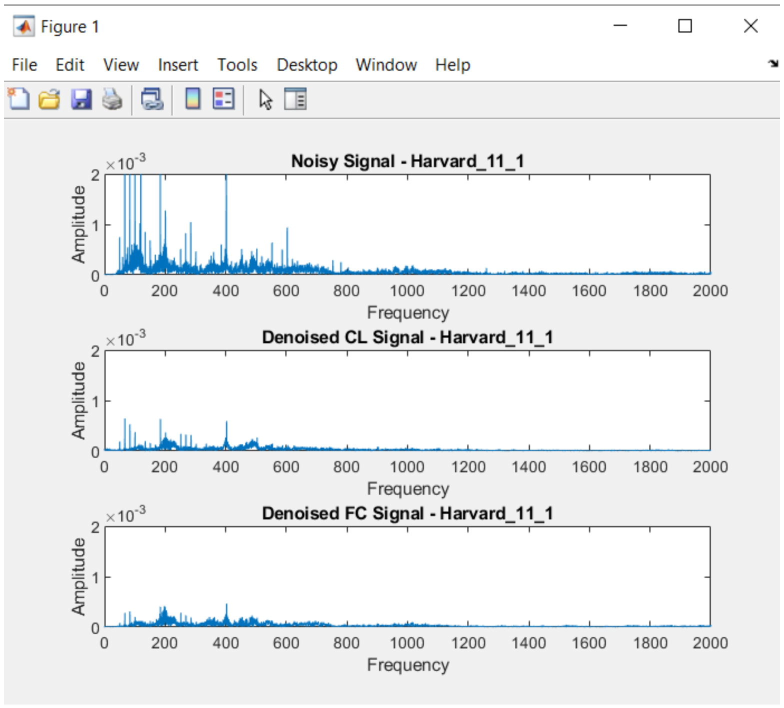


Figure 28: Frequency Representation of Original Signal and FC and CL Denoised Signals

It is through these two figures that the different denoising mechanisms can be seen. In the filtered signals, it is clear that frequencies above a certain prescribed value were simply attenuated, regardless of what they contained. In the plots that related to deep-learning denoising, this did not occur. For both the CL and the FC signal, it can be seen that frequencies are not eliminated above a certain threshold, rather noise is processed out of the system across the entire spectrum based on whether the algorithm detects it as speech or not. This is likely why the filtered signals came out more distorted and muffled than the deep learning signals. The deep learning algorithm is specifically trained to segregate speech from noise and only removes noise values from the signal. The filtering process is a bit more rudimentary and is subject to the values the user selects, which may exclude necessary information from higher frequencies.

Next, utilizing the VAD on each technique, we can verify that speech was properly removed. Referring to Figures 9, 10, and 11 above, it can be seen that speech within the signal was covered up by so much noise that the VAD could not distinguish between the two. If the noise was properly removed, it would be expected to see more peaks and valleys on the plot, the probability only reaching 1.0 when speech was actively present. In order to verify this assumption, the MATLAB voice activity detector was implemented on a total of twelve samples

[19]. This included each denoising technique (FIR, IIR, CL, and FC) for the three standard audio files used throughout the report. For the sake of simplicity, only the results of the harvard_11_1 audio file will be shown directly, but the other figures will be included in the attached documentation. See Figures 29, 30, 31, and 32 below for results.

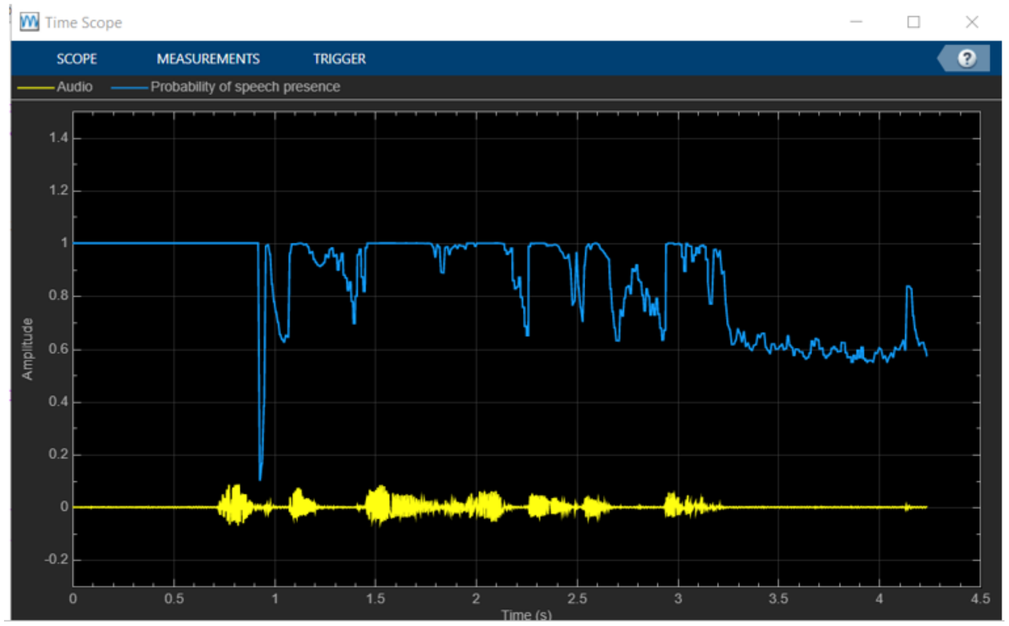


Figure 29: VAD Implementation of FIR Filtered Audio

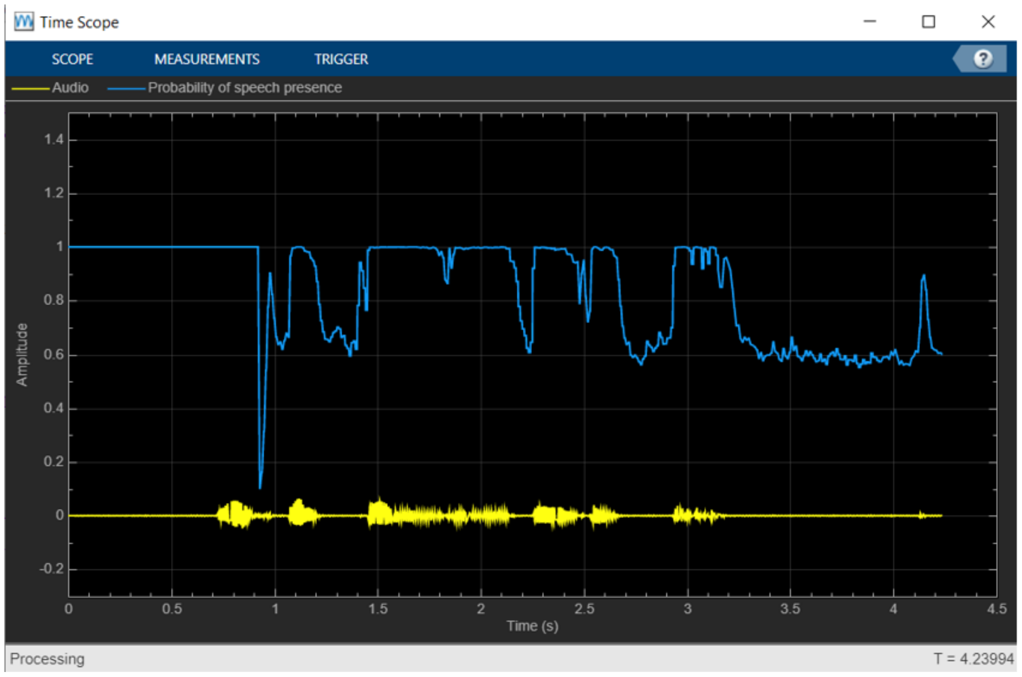


Figure 30: VAD Implementation of IIR Filtered Audio

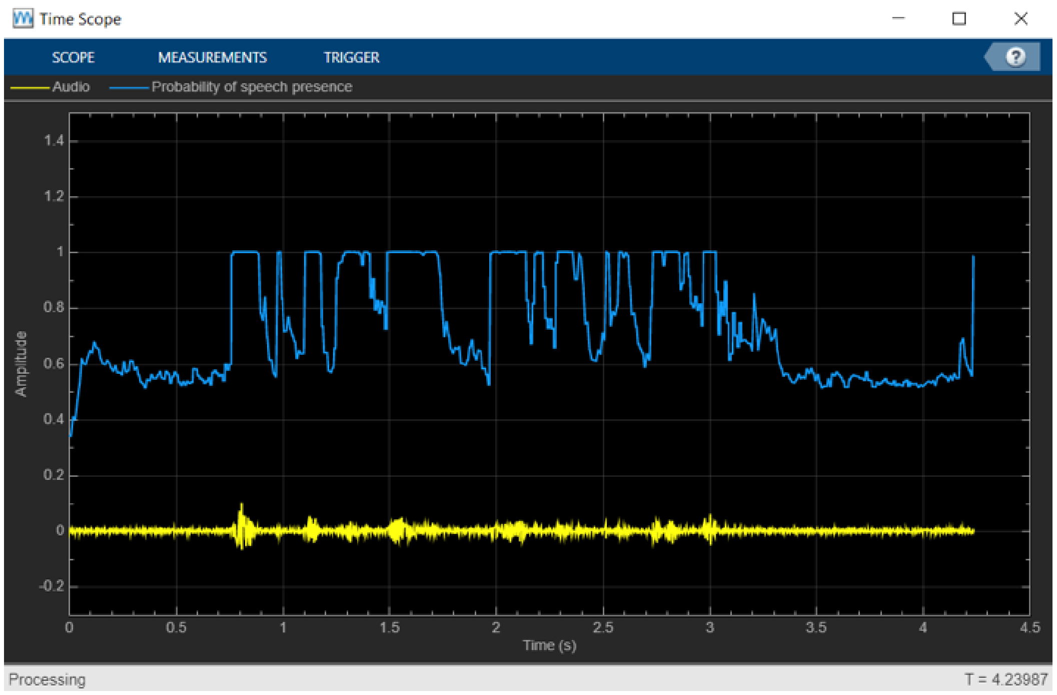


Figure 31: VAD Implementation of CL Deep Learning Denoised Audio

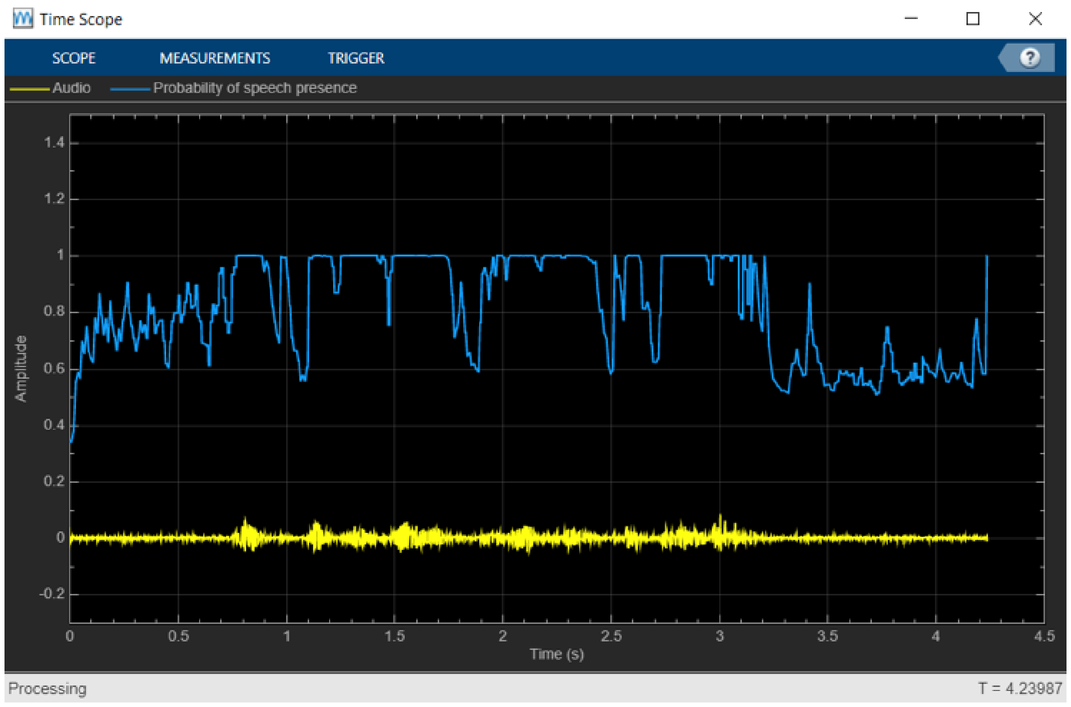


Figure 32: VAD Implementation of FC Deep Learning Denoised Audio

When all of these plots are compared to the original, it is clear that the noise reduction worked to some extent utilizing every one of the techniques. When the original signal was processed through the VAD, there was much less consistency and the probability was seemingly random for much of the signal. However, in the plots of the post-processed signals, there is an increase in the number of consistent and flat peaks within the probability curve, which matched up with the peaks in the audio signal. This means that the VAD appeared to be registering speech, and only speech, at values where it was present. The peaks in which speech was active were in line with expectations and remained at a high probability over the duration of the system measurements. On the original signals, the peaks were inconsistent and rapidly varying due to the noise present. Finally, the time domain plots generated by the VAD after processing are significantly less convoluted.

In terms of a comparison between the deep-learning and filtering methods, it appears as if the deep-learning algorithm was more successful at removing noise than the filtering methods. Although it is highlighted specifically on the CL plot, there is a much larger number of peaks and valleys that line up with active speech on both deep-learning graphs as opposed to the filtering methods. While the filtering plots are consistent, they appear to measure a high probability during frames in which speech should not be present. Overall, while both denoising methods proved to be successful to some extent, it was clear that the deep-learning method was more apt at completely segregating and removing unwanted noise from an audio signal.

VII. Conclusion:

In conclusion, this report analyzed three various audio processing techniques that are utilized throughout industry today: voice activity detection (VAD), filtering, and deep-learning mechanisms. Unfortunately, due to issues operating the voice activity toolbox provided by MATLAB, we were unable to implement that method directly into the presented research. It was, however, useful for verification purposes as it provides useful data for the comparison of processed and pre-processed signals. In terms of the other two methods, filtering and deep-learning, the group found various levels of success. The filtering technique successfully removed noise from the audio signal, however, it also removed key frequencies from the speech frames, resulting in a muffled and distorted output audio. In contrast to that, the deep-learning method was more successful in removing noise from the input signal while maintaining the integrity of the speech frames. Since the deep-learning method segregates noise from speech automatically, the speech frames were not suppressed in any form. While there was some distortion, it was quite minimal compared to that of the filtered signals. Perhaps with some tuning and altering of parameters, the filtering method could reach the same quality of the deep-learning process, however, that assumption would require additional research and falls outside the scope of this report. For a full collection of all data and MATLAB scripts used, please refer to the provided GitHub link [20].

References

[1] G. Sharma, K. Umapathy, S. Krishnan, "Trends in Audio Signal Feature Extraction Methods," Applied Acoustics, Vol. 158. Doi: 10.1016/j.apacoust.2019.107020. https://www.sciencedirect.com/science/article/pii/S0003682X19308795

[2] R.G. Bachu, S. Kopparthi, B. Adapa, B.D. Barkana, "Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal," University of Bridgeport School of Engineering. https://www.asee.org/documents/zones/zone1/2008/student/ASEE12008_0044_paper.pdf

[3] R. Hasan, M. Jamil, G. Rabbani, S. Rahman, "Speaker Identification Using Mel Frequency Cepstral Coefficients," 3rd International Conference on Electrical and Computer Engineering, Dhaka Bangladesh, 2004, pp. 28-20. https://www.researchgate.net/profile/Golam-Rabbani/publication/255574793_Speaker_Identification_Using_Mel_Frequency_Cepstral_Coefficients/links/55f05d5908ae0af8ee1d1894/Speaker-Identification-Using-Mel-Frequency-Cepstral-Coefficients.pdf

[4]T. H. Applebaum and B. A. Hanson, "Regression features for recognition of speech in quiet and in noise," [Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing, Toronto, ON, Canada, 1991, pp. 985-988 vol.2, doi: 10.1109/ICASSP.1991.150506. https://ieeexplore-ieee-org.ezproxy.neu.edu/stamp/stamp.jsp?tp=&arnumber=150506&tag=1

[5] K. C. Jayashree, Karthik M. Bharamappa, P .Manoj Kumar, G. Mahaa Santeep, and Sugandha Saxena. (2020). Denoising Real-Time Audio Signals Using Matlab Filter Techniques. https://easychair.org/publications/preprint_download/Bqmp

[6] D. S. Jat, A. S. Limbo, and C. Singh, "Chapter 6 - Voice Activity Detection-Based Home Automation System for People With Special Needs," in *Intelligent speech signal processing*, N. Dey, Ed. London: Elsevier, 2019, pp. 101–111.

[7] I. Hwang and J. H. Chang, "Voice Activity Detection Based on Statistical Model Employing Deep Neural Network," 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kitakyushu, Japan, 2014, pp. 582-585, doi: 10.1109/IIH-MSP.2014.150.

[8] A. d. S. P. Soares et al., "Energy-based voice activity detection algorithm using Gaussian and Cauchy kernels," 2018 IEEE 9th Latin American Symposium on Circuits & Systems (LASCAS), Puerto Vallarta, Mexico, 2018, pp. 1-4, doi: 10.1109/LASCAS.2018.8399936. https://ieeexplore.ieee.org/document/8278160

[9] F. Muzaffar, B. Mohsin, F. Naz and F. Jawed, "DSP Implementation of Voice Recognition Using Dynamic Time Warping Algorithm," 2005 Student Conference on Engineering Sciences and Technology, Karachi, Pakistan, 2005, pp. 1-7, doi: 10.1109/SCONEST.2005.4382877.

[10] Krishnakumar, Harshit & Williamson, Donald. (2019). A Comparison of Boosted Deep Neural Networks for Voice Activity Detection. 1-5. 10.1109/GlobalSIP45357.2019.8969258.

[11] A. Sehgal and N. Kehtarnavaz, "A Convolutional Neural Network Smartphone App for Real-Time Voice Activity Detection," in IEEE Access, vol. 6, pp. 9017-9026, 2018, doi: 10.1109/ACCESS.2018.2800728. https://ieeexplore.ieee.org/document/6998396

[12] C. Shi, N. Jiang, H. Li, “On Algorithms and Implementations of a 4-Channel Active Noise Canceling Window,” 2017 International Symposium on Intelligent Signal Processing and Communications Systems, 2017. https://ieeexplore-ieee-org.ezproxy.neu.edu/stamp/stamp.jsp?tp=&arnumber=8266476&tag=1

[13] “IEEE Recommended Practice for Speech Quality Measurements,” IEEE Transactions on Audio and Electroacoustics, vol. 17, no. 3. 1969. doi: 10.1109/TAU.1969.1162058. https://ieeexplore.ieee.org/document/1162058

[14] R. Martin. “Noise Power Spectral Density Estimation Based on Optimal Smoothing and Minimum Statistics,” IEEE Transactions on Speech and Audio Processing. vol. 9, no. 5, 2001. Doi: 10.1109/89.928915. https://ieeexplore.ieee.org/document/928915

[15] Y. Ephraim, D. Malah, “Speech Enhancement Using a Minimum-Mean Square Error Short-Time Spectral Amplitude Estimator,” IEEE Transactions on Acoustics, Speech, and Signal Processing. vol. 32, no. 6, 1984. doi: 10.1109/TASSP.1984.1164453.
https://ieeexplore.ieee.org/document/1164453

[16] J. Sohn, N.S. Kim, W. Sung, “A Statistical Model-Based Voice Activity Detection,” IEEE Signal Processing Letters, vol. 6, no. 1, 1999. doi: 10.1109/97.736233.
https://ieeexplore.ieee.org/document/736233

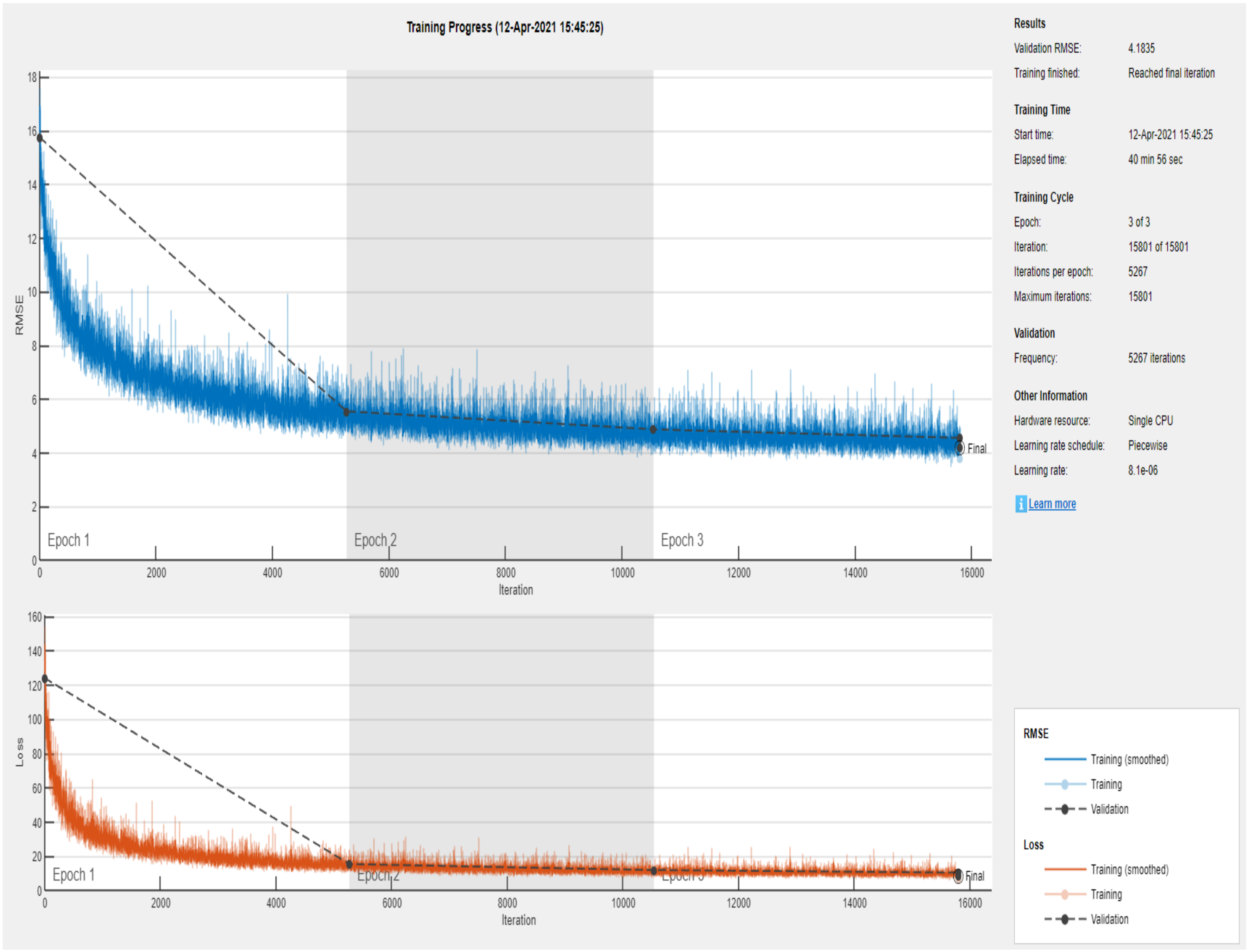
[17] “Factors Influencing Fundamental Frequency,” The National Center for Voice and Speech, Tutorials - Voice Production.

[18]“Denoise Speech Using Deep Learning Networks,” *MATLAB & Simulink*, 2021. [Online]. Available:
https://www.mathworks.com/help/deeplearning/ug/denoise-speech-using-deep-learning-networks.html

[19] “Detect Presence of Speech in Audio Signal,” *MATLAB & Simulink*, 2021. [Online]. Available: https://www.mathworks.com/help/audio/ref/voiceactivitydetector-system-object.html

[20] K. Yu, N. Guidoboni, and E. Marcello, “EECE5666_SP2021,” *GitHub*, 28-Apr-2021. [Online]. Available: https://github.com/ItsKYu/EECE5666_SP2021. [Accessed: 28-Apr-2021].

APPENDIX



Results

Validation RMSE:	4.1835
Training finished:	Reached final iteration

Training Time

Start time:	12-Apr-2021 15:45:25
Elapsed time:	40 min 56 sec

Training Cycle

Epoch:	3 of 3
Iteration:	15801 of 15801
Iterations per epoch:	5267
Maximum iterations:	15801

Validation

Frequency:	5267 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Piecewise
Learning rate:	8.1e-06