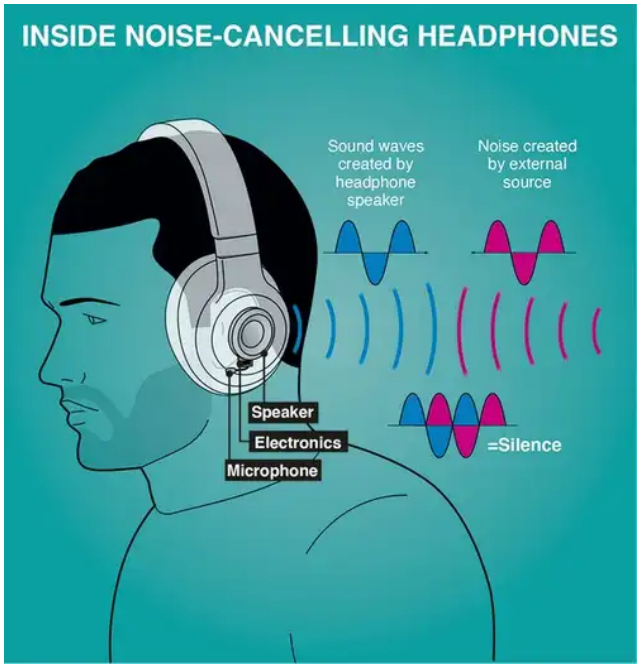
Eklavya

AI NOISE REDUCTION

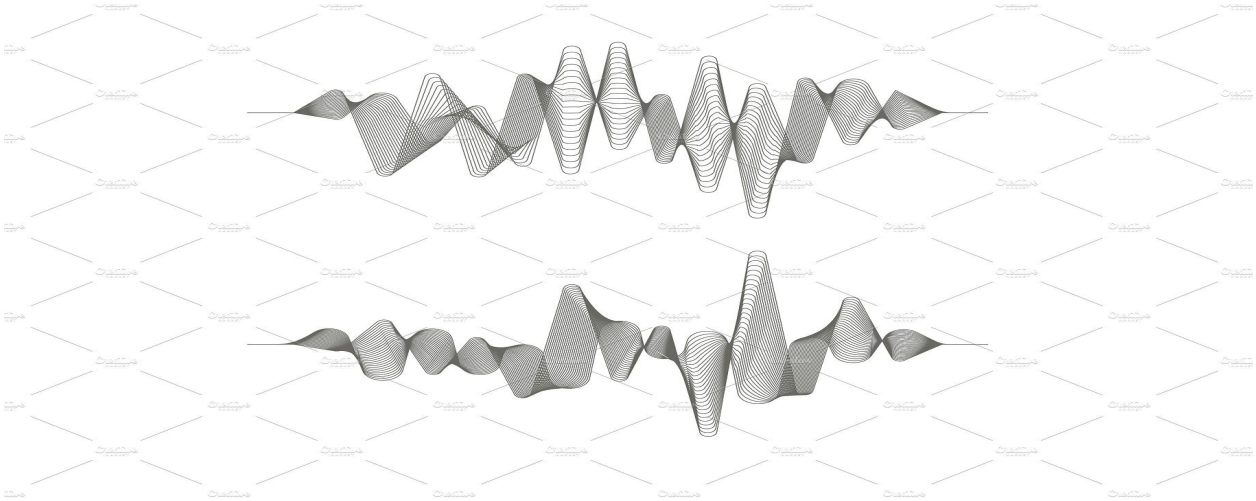


When some annoying noise springs up and you have put on your headphones that are cheap and don't offer Active Noise Cancellation , how do you feel ? Definitely unpleasant. And the problem is that you can't take any action because you don't know when that annoying sound will reappear.Noise. It's all around us. Whether it's a crying baby, the gossiping of colleagues or even the whistling of a pressure cooker. A quiet place to work or enjoy some relaxation is becoming a truly premium commodity. Noise cancellation technology can help bring the zen back into your life. Automation of reducing audio noise is necessary even though we don't have costly headphones. As engineers we are bound to do something about it.

A Basic overview of *SOUND* :

Sound is energy travelling as a wave through some form of matter. We generally hear sound transmitted through the medium which is air.

For example, a guitar string vibrates in air, knocking air molecules about. Those molecules knock into their neighboring molecules, and so on.We are known to terms like **Compression and Rarefaction.** When the wave moving through the air hits our eardrums, it moves them at the same frequency as the guitar string.



Our brain converts that movement into electrical signals that you then perceive as sound. That’s a very simplistic explanation of what sound is, but it’s the least you need to know to understand noise cancellation.

The algorithm is used to calculate the anti-noise wave needed to destructively interfere with the ambient noise where you are. Then a device known as a transducer, which is essentially a type of speaker, generates the anti-noise wave.

The end effect is that when you switch on the noise cancellation function, you're suddenly enveloped in (near) silence. Which allows you to enjoy your audio at much better fidelity or just enjoy some peace and quiet.

I know you are wondering that the name of Article AI Noise Cancellation ,then where is AI .So here it is :

AI-Powered Noise Removal

A fairly new development is the application of artificial intelligence technology to remove unwanted noise from an audio signal. Unfortunately this can't quite happen in real time just yet, so it isn't effective at noise cancellation. What it is very good at is removing noise from a microphone signal.

For example, if you're trying to Skype someone or make a voice recording in a noisy environment, it can be hard for the person on the other end to understand what you're saying. Using artificial intelligence, that audio stream can be analyzed and everything but your own voice can be removed.

In this Part I of our series we discuss methods of how it's done formally.

Introduction

To implement this Algorithm we will require :

1.A *signal* audio clip containing the signal and the noise intended to be removed.

2.Knack to learn some concepts related to signal processing.

Process:

We have a signal that we can extract some information from.We can extract data regarding amplitude,frequency,etc of the signal to convert it into the required range.

```
wav_loc = "assets/audio/fish.wav"

rate, data = wavfile.read(wav_loc)

data = data / 32768
```

How do we identify noise? Here is the answer: we are setting some threshold values so that the values outside are removed as noise.

.PCM(Pulse-Code Modulation), .wav(Waveform Audio File), .AIFF(Audio Interchange File Format),MP3, etc are some different types of files

We need to do the following to remove this(.wav) kind of "noise" :

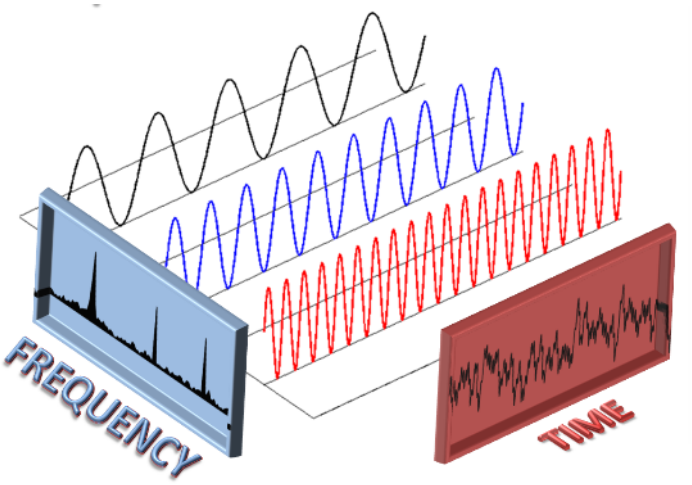
- Calculate the threshold for each frequency
- Mask if the signal is above the threshold
- Convolve the mask with the smoothing filter.
- After all these operations,the signal is recovered using abs function.

Some fundamental mathematical tools we need to understand (transforms → fourier transform) are - knowledge of complex numbers,Integration,some basic sine cos functions.

We are concerned about the frequency domain-As the purpose of the project itself is to remove noise, that is a signal whose frequency is not in required range.Also working in frequency domain is quite easier than that of time and space.

For the above purpose ,we have to use a transform method (Fourier Transform).To get clear idea of this from this site Fourier Transforms (tutorialspoint.com).
After taking the Fourier transform,the function is decomposed into the function depending on frequency rather than depending on time and space.The **Fourier Transform** is a tool that breaks a waveform (a function or signal) into an alternate representation, characterized by sine and cosines. The **Fourier Transform** shows that any waveform can be re-written as the sum of sinusoidal functions.The following image

will help us to understand the terminology to a greater extent.Here,we are dividing the complex time signal into simple discrete frequency signals.



To reduce the consumption of time ,we need to have a faster mechanism than Fourier Transform.So here it is,the FFT(Fast Fourier Transform).The Fast Fourier Transform (FFT) is a way to reduce the complexity of the Fourier transform computation from O(n2)O(n2) to O(nlogn)O(nlogn), which is a dramatic improvement.

Let's take an example to have a stronger grip on the topic.

Suppose we have a time series y1,y2,....,yn and we want to compute the complex Fourier coefficient z1. Going by the formula in the previous section, this would require computing

$$z_0 = \sum_{t=0}^{n-1} y_t,$$

which is simply proportional to the mean of the data. If the data have been de-meaned or de-trended then this will be zero. The next Fourier coefficient is then

$$\begin{aligned} z_1 &= \sum_{t=0}^{n-1} y_t \exp(-2\pi i \cdot 1 \cdot t/n) \\ &= y_0 \exp(-2\pi i \cdot 1 \cdot 0/n) + y_1 \exp(-2\pi i \cdot 1 \cdot 1/n) + y_2 \exp(-2\pi i \cdot 1 \cdot 2/n) + \cdots. \end{aligned}$$

Now suppose we want to compute the next coefficient z_2. This requires computing

$$z_2 = y_0 \exp(-2\pi i \cdot 2 \cdot 0/n) + y_1 \exp(-2\pi i \cdot 2 \cdot 1/n) + \cdots$$

But wait! Notice how the exponential in the second term in the sum for z2 is the same as the exponential in the *third* term in the sum for z1. They are both equal to exp(−2πi·1·2/n). There is no need to compute this exponential quantity twice. We can simply compute it the first time, store it in memory, and then retrieve it when it is needed to compute
z2(assuming that retrieving from memory is faster than computing it from scratch.) One can think of the FFT algorithm as an elaborate bookkeeping algorithm that keeps track of these symmetries in computing the Fourier coefficients.
Suppose we have a series y1,y2,....,yn and we want to compute the complex Fourier coefficient z1. Going by the formula in the previous section, this would require computing.
Now ,the key function of the project (subtract/reduce noise):

Algorithm to separate / subtract / reduce noise

- After adding the noise signal ,it is to be removed to achieve the result.
- To separate/subtract ,we are defining a threshold value
- And then a remove function is used for masking,filtering ,etc.

```
# mask if the signal is above the threshold
```

```
    sig_mask = sig_stft_db < db_thresh

    if verbose:

        print("Masking:", td(seconds=time.time() -
start))

        start = time.time()
```

What else we can do is : We can improve if the model learns by itself and adapts the various surroundings and removes the various noises, may it be environmental ,industrial or any. But it is quite more than the scope of this blog.Hope,we will discuss it later ;)

Now,in Part || ,we will go sequentially for further understanding .

The basic idea of the **FFT** is to apply divide and conquer. We divide the coefficient vector of the polynomial into two vectors, recursively compute the DFT(Discrete Fourier Transform) for each of them, and combine the results to compute the DFT of the complete polynomial. Simple meaning is to reduce the complexity of the function we use FFT.Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

Here,we are creating a function having a parameter,which is to be assigned the array of data type complexes. Also defining phases using random functions in python.Then after converting phases into complex form like a+jb ,the real part is returned.

Now,we are setting some minimum and maximum value of frequency, sample and sample rate to get the generated noise in the required frequency range .Then by setting frequency range, we will add this generated noise into our original data signal to output

a signal containing noise which we have to remove in further implementation.Also convert noise in dB.

To denoise the signal we are using a remove function having various parameters having some conditions to check for threshold ,application,recovery,etc.It masks(**process where one sound is rendered inaudible because of the presence of another sound**) if the signal is above the threshold ,convolve the mask if with a smoothing filter.We can also plot spectrograms of these processes. There is method time.time to calculate the required time by subtracting the initial one.After returning from the remove function ,the noise will be reduced.

Result :

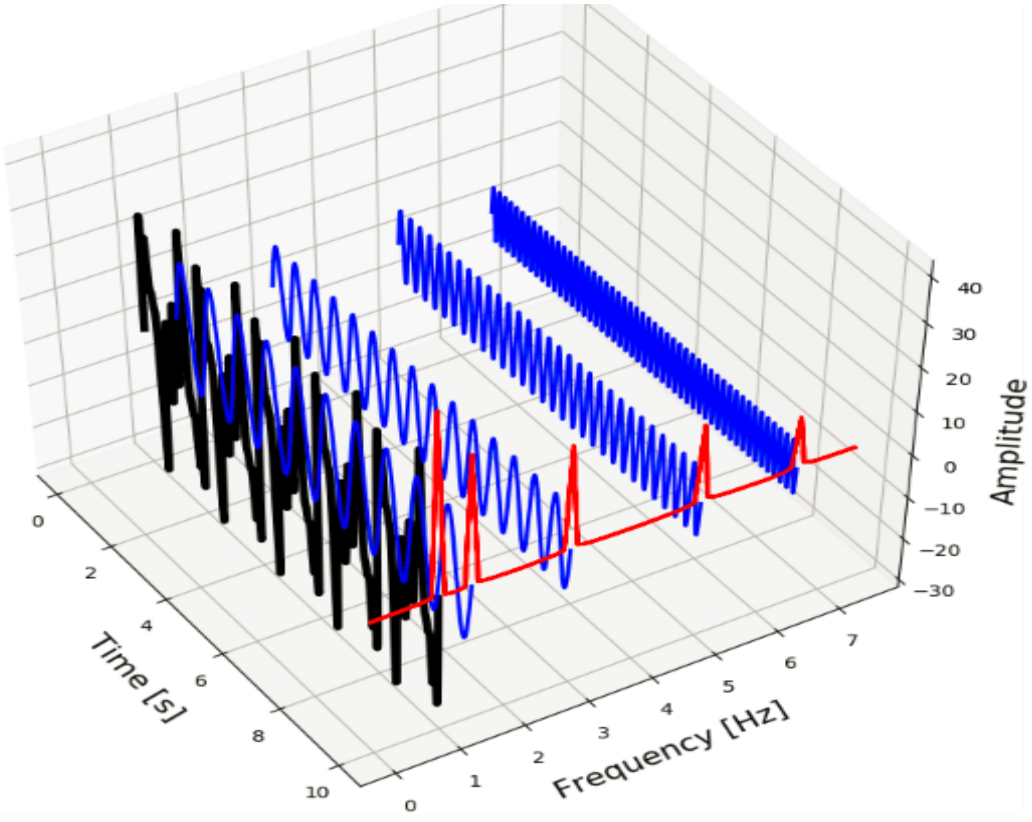
The key idea behind the whole Algorithm is to create a signal which is out of phase with the Noise signal.Now,the AI Noise reduction algorithm is ready to be implemented with the coding practice.Enjoy the uninterrupted conversations calmly.

---------- Thank You :) ----------

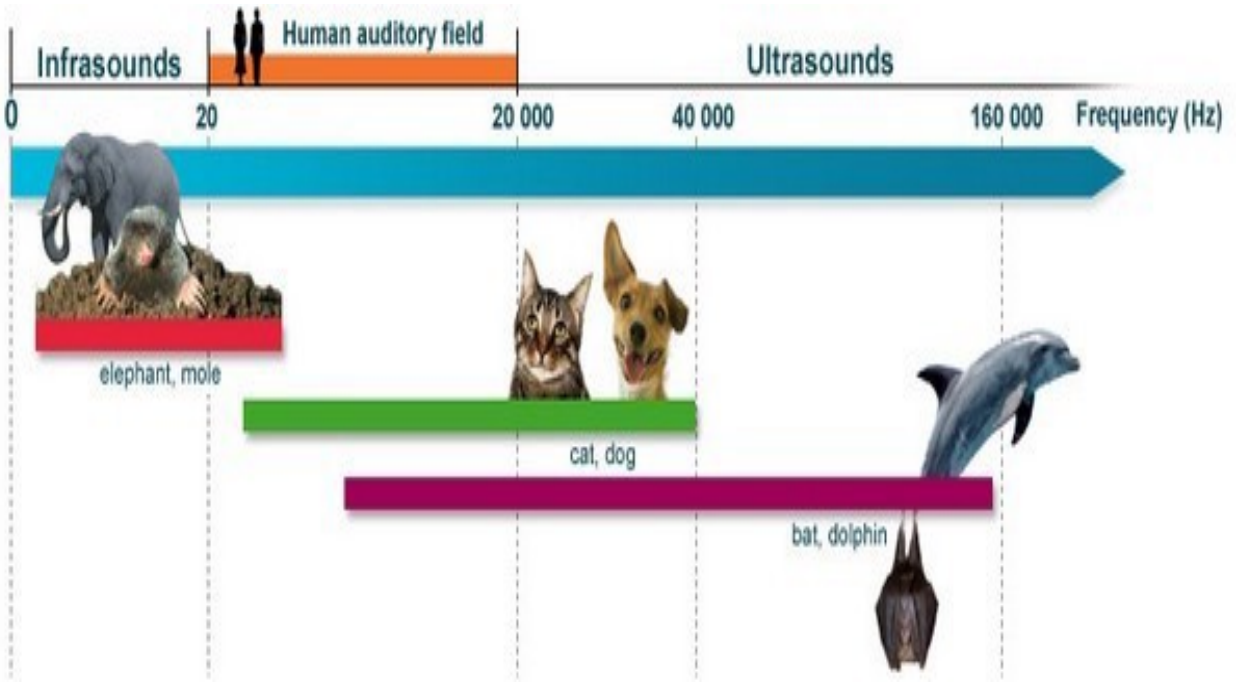
★ **We can refer to following questions for more details :**

1 .We have a signal that we can extract some information from. But what information can we get?

Audio waves are the vibration of air molecules whenever any sound happens and sound travels from originator to the receiver in the form of wave.
As such, this wave has 3 properties to it — Amplitude, Frequency and Time.
→ Amplitude represents the magnitude of the wave signal and it is usually measured in decibels (dB).
→ Time is the time scale, as we all know it.
→ Frequency represents how many complete cycle the wave takes in one second and it is measured in Hz.



Every living being has a different hearing range of sound wave.



We (humans) could hear sound waves that are in the range of 20 Hz to 20,000 Hz. Dogs could hear the sound waves upto 45K Hz and dolphins could hear until 150K Hz.
As human hearing range is around 20K Hz, sampling rate of audio files in many libraries are by default set at 22050 per sec. We do have an option of increasing/decreasing it as per the need.
Note audio waves are continuous in nature — but most of our processing engines are built to process digital/discrete signals.

```
wav_loc = r'/home/dhriti/Downloads/wave/download.wav' # giving the file location
rate, data = wavfile.read(wav_loc)    # reading the wav file
data = data / 32768  #-32767 to +32767 is proper audio (to be symmetrical) and 32768 means that the audio clipped at that point
#wav-file is 16 bit integer, the range is [-32768, 32767], thus dividing by 32768 (2^15) will give the proper two's-complement range of [-1, 1]
```

2.How do we identify noise?

The noise identification technology used in Conversa analyzes and classifies sounds by simultaneously examining three specific characteristics inherent in the signals:

- *Magnetic deviation*
- *Change in the pressure*
- *Intensity Change*: Defined as the change in the intensity of the audio signal over a selected time period.
- *Modulation Frequency*: Defined as the rate at which the signal's intensity changes over a selected time period.
- *Time*: Defined as the duration of the signal.

Using these three dimensions, the four signal types (three types of noise plus desirable signals) can be placed on the continuum scale found in Table 1.

	Stationary	Pseudo Stationary	Desirable	Transient
Intensity Changes	smallest < ---------- > largest			
Modulation Frequency	lowest < ---------- > highest			
Time Duration	longest < ---------- > shortest			

TABLE 1. A continuum exists between the three specific characteristics found in signals (ie, intensity changes, modulation frequency, and duration) and the four signal types (ie, stationary, pseudo-stationary, desirable, and transient) .

Thus by checking with their characteristics we can match with the predefined noise and thus identify the given noise .

An **error which is superimposed on top of a true signal**. Noise may be random or systematic. Noise can be greatly reduced by transmitting signals digitally instead of in analog form because each piece of information is allowed only discrete values which are spaced farther apart than the contribution due to noise.

3.Different kinds of noise?

The different types of noise include **physical, semantic, psychological, and physiological**. Each interferes with the process of communication in different ways.

- OpenSimplex noise.
- Perlin noise.
- Simplex noise.
- Simulation noise.
- Wavelet noise.
- White noise. Additive white Gaussian noise.
- Black noise.
- Gaussian noise.
- Pink noise or flicker noise, with 1/f power spectrum.
- Brownian noise, with $1/f^2$ power spectrum.(Fractal Noise)
- Contaminated Gaussian noise, whose PDF is a linear mixture of Gaussian PDFs.
- Power-law noise.
- Cauchy noise.
- Periodic Noise. …
- Impulse Valued Noise (Salt and Pepper Noise) ...
- Quantization noise.
- https://arxiv.org/pdf/1505.03489.pdf

Signal processing noise can be classified by its statistical properties (sometimes called the "color" of the noise) and by how it modifies the intended signal:

- Multiplicative noise, multiplies or modulates the intended signal
- Quantization error, due to conversion from continuous to discrete values
- Poisson noise, typical of signals that are rates of discrete events
- Shot noise, e.g. caused by static electricity discharge
- Transient noise, a short pulse followed by decaying oscillations
- Burst noise, powerful but only during short intervals
- Phase noise, random time shifts in a signal

What do we need to do to remove “this” kind of “noise”?

Artificial neural networks are an old idea that have recently exploded in the form of deep learning. While there are different deep learning approaches to noise removal, they all work by learning from a training dataset.

Training Datasets for Noise Removal

The first step to building an accurate noise removal model is to construct a quality training dataset. Since our goal is to remove background noise, our dataset should consist of recordings of clean speech paired with its noisy variant.

Before assembling a dataset, it is important to consider the use case of the model. For example, when training a noise removal algorithm that would be applied to signals from a helicopter pilot’s microphone, it makes most sense to train the network with audio samples that are distorted by variations of helicopter sounds. For a general use noise removal model, it makes sense to train with samples of everyday background sounds such as loud chatter, air conditioning, typing, dogs barking, traffic, music — you get the idea.

Once we’ve figured out what kind of data we want to train with, we have to actually generate the data set. The best way is to find a large amount of clean speech signals and pure noisy signals and combine them in all sorts of ways. For example, you can combine a high quality sample of a person speaking and a sample of a dog barking to produce a new sample which would have a person speaking with barking in the background. So, by providing both the original sample of the person speaking and sample with both the speech and the dog barking, the neural network can repeatedly

compare its estimated clean speech signal to the actual clean speech signal to then adjust itself and try again.

Finally, we can now feed our dataset to a neural network, so it can learn to isolate the background noise and generate clean speech. One of the most popular and effective for audio processing is the Recurrent Neural Network.

Reference link
https://towardsdatascience.com/background-noise-removal-traditional-vs-ai-algorithms-9e7ec5776173