Application of different filters on audio signals for noise reduction

YOGESH DEWANGAN (204102319)

April 21, 2021

1 Abstract

Noise reduction is the process of removing noise from a signal. Noise reduction techniques exist for audio and images. Noise reduction algorithms may distort the signal to some degree.All signal processing devices, both analog and digital, have traits that make them susceptible to noise. Noise can be random or white noise with an even frequency distribution, or frequency-dependent noise introduced by a device's mechanism or signal processing algorithms. We have different types of filters for removing the different types of noises. it's not necessary that all filter works good for all the noises. Here we are going to analyse the different filters on different types of noises, and showing their performance based on Signal to Noise ratio.
Source - https://en.wikipedia.org/wiki/Noise_reduction

2 Introduction

Over the past few decades, the problem of controlling noise level in the environment especially in the audio processing field has been the focus of a tremendous amount of research. Many papers have dealt with noise reduction in audio applications to improve the quality of audio signals.

2.1 Different types of Noises:

1) White Noise = In signal processing, white noise is a random signal having equal intensity at different frequencies, giving it a constant power spectral density.

2) Poission Noise: Shot noise or Poisson noise is a type of noise which can be modeled by a Poisson process. In electronics shot noise originates from the discrete nature of electric charge. Shot noise also occurs in photon counting in optical devices, where shot noise is associated with the particle nature of light.

3 Methodology

In this project we are using three different approach for noise reduction :

(a) Wiener filter
(b) Haar wavelet
(c) Thresholding

3.1 Wiener filter:

In signal processing, the Wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant (LTI) filtering of an observed noisy process, assuming known stationary signal and noise spectra, and additive noise. The Wiener filter minimizes the mean square error between the estimated random process and the desired process.
Assume: x(n) is the original signal, y(n) is the distorted signal.

The goal here is an approximation of the original signal x(n) in the least mean squared sense, meaning we would like to **minimize the mean quadratic error** between the filtered and the original signal.

We have a filter system with Wiener Filter $h_W(n)$:

$$x(n) = y(n) * h_W(n)$$

meaning we filter our distorted signal y(n) with our still unknown filter $h_W(n)$.

The convolution $h_W(n)$ of (with filter length L) with y(n) can be written as a matrix multiplication:

$$x(n) = \sum_{m=0}^{L-1} y(n-m) \cdot h_W(m)$$

Alternatively, we can run the sum in the other direction,

$$x(n) = \sum_{m=0}^{L-1} y(n-L+1+m) \cdot h_W(L-1-m)$$

Now let's define 2 vectors. The first is a vector of the **past L samples of our noisy signal y**, up to the present sample at time n, (bold face font to indicate that it is a vector)

$$\boldsymbol{y}(n) = [y(n-L+1), \dots, y(n-1), y(n)]$$

The next vector contains the **time reversed impulse response**,

$$\boldsymbol{h_W} = [h_W(L-1), \dots, h_W(1), h_W(0)]$$

Using those 2 vectors, we can rewrite our convolution equation above as a vector multiplication,

$$x(n) = \boldsymbol{y}(n) \cdot \boldsymbol{h_W^T}$$

Observe that $\boldsymbol{h_W}$ has no time index because it already contains all the samples of the time-reversed impulse response, and is constant.

We can now also put the output signal x(n) into the **row vector**,

$$\boldsymbol{x} = [x(0), x(1), \dots]$$

To obtain this column vector, we simply assemble all the row vectors of our noisy signal $\boldsymbol{y}(n)$ into a matrix $\boldsymbol{A}$,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{y}(0) \\ \boldsymbol{y}(1) \\ \vdots \end{bmatrix}$$

With this matrix, we obtain the result of our convolution at all time steps n to

$$\boldsymbol{A} \cdot \boldsymbol{h_W^T} = \boldsymbol{x}^T$$

this is just another way of writing our convolution.

For the example of a filter length of $\boldsymbol{h_W}$ of L=2 hence we get,

$$\begin{bmatrix} y(0) & y(1) \\ y(1) & y(2) \\ y(2) & y(3) \\ \vdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} h_W(1) \\ h_W(0) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \end{bmatrix}$$

Observe again that the vector $\boldsymbol{h}_w$ in this equation is the time reversed impulse response of our filter. This is now the **matrix multiplication** formulation of our **convolution**.

We can now obtain the minimum mean squared error **solution** of this matrix multiplication using the so-called Moore-Penrose **Pseudo Inverse**

This pseudo-inverse finds the column vector $\boldsymbol{h}^T$ which minimizes the distance to a given $\boldsymbol{x}$ with the matrix $\boldsymbol{A}$ (which contains our signal to be filtered):

$$\boldsymbol{A} \cdot \boldsymbol{h_W^T} \rightarrow \boldsymbol{x}^T$$

Matrix $\boldsymbol{A}$ and vector $\boldsymbol{x}$ are known (this is done in a **"trainings"-phase** to obtain the Wiener filter coefficients $\boldsymbol{h_W}$, from noisy signals in matrix **A** and clean signals in vector **x**), and vector $\boldsymbol{h}_w$ is unknown so far. After the trainings-phase the filter can also be applied to **similar signals**.

This problem can be solved exactly if the matrix $\boldsymbol{A}$ is **square and invertible**. Just multiplying the equation with $\boldsymbol{A}^{-1}$ from the left would give us the solution

$$\boldsymbol{h_W^T} = \boldsymbol{A}^{-1} \cdot \boldsymbol{x}^T$$

This cannot be done, if $\boldsymbol{A}$ is **non-square**, for instance if it has many more rows than columns. In this case we don't have an exact solution, but many solutions that come close to $\boldsymbol{x}$. We would like to obtain the solution which comes **closest** to $\boldsymbol{x}$ in a mean squared error distance sense (also called **Euclidean Distance**).

This solution is derived using the Pseudo-Inverse:

$$\boldsymbol{A}^T \cdot \boldsymbol{A} \cdot \boldsymbol{h_W^T} = \boldsymbol{A}^T \cdot \boldsymbol{x}^T$$

Here, $\boldsymbol{A}^T \cdot \boldsymbol{A}$ is now a square matrix, and usually invertible, such that we obtain our solution

$$\boldsymbol{h_W^T} = (\boldsymbol{A}^T \cdot \boldsymbol{A})^{-1} \boldsymbol{A}^T \cdot \boldsymbol{x}^T$$

This $\boldsymbol{h}_w$ is now the **solution** we where looking for. This solution has the minimum mean squared distance to the un-noisy version of all solutions.

3.2 Thresholding:

Threshold means setting the limit for the passing the signal. After setting the threshold we can remove the upper or lower part of the signal. For stting the threshold values we are using the FFT technique.
Fast Fourier Transformation(FFT) is a mathematical algorithm that calculates Discrete Fourier Transform(DFT) of a given sequence. The only difference between FT(Fourier Transform) and FFT is that FT considers a continuous signal while FFT takes a discrete signal as input. DFT converts a sequence into its frequency constituents just like FT does for a continuous signal. In our case, we have a sequence of amplitudes that were sampled from a continuous audio signal. FFT turns time domain into frequency domain which let's us identify the ingredients of our smoothie.
there is quite the noise in our FFT result. But we can still identify three peaks in the FFT frequency magnitude chart, also called periodograms. We can use what is called a threshold noise filter and filter the noise out by only accepting those frequencies whose magnitudes exceed the threshold of the given quantile. Zero out all frequencies below that quantile, take the inverse FFT of it becomes the following graph.

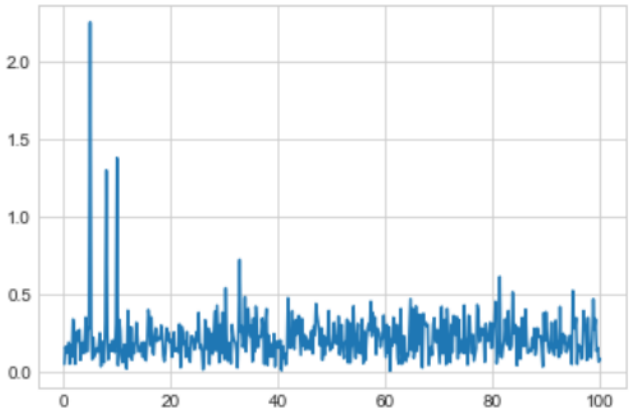


Figure: Result of FFT with noise

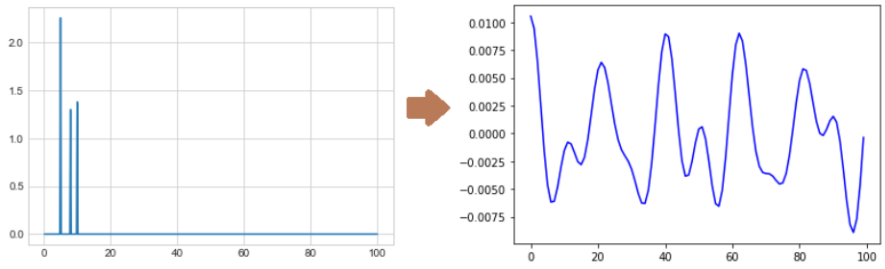


Figure: Cleared noise and the inverse FFT.

3.3 HaarWavelet:

The wavelet transform (WT) has broad application in the analysis of stationary and nonstationary signals. These applications include the removal of electrical noise from the signals, detection of abrupt discontinuities, and compression of large amounts of data.
With the WT, it is possible to decompose a signal into a group of constituent signals, known as wavelets, each with a well defined, dominant frequency, similar to the Fourier transform (FT) in which the representation of a signal is by sine and cosine functions of unlimited duration. In WT, wavelets are transient functions of short duration, that is, limited duration centered around a specific time. The problem of the FT is that when passing from the time domain to the frequency domain the information of what happens in time is lost. Observing the frequency spectrum obtained using the FT is simple to distinguish the frequency content of the signal being analyzed but it is not possible to deduce in what time the components of the signal of the frequency spectrum appear or disappear. Unlike the FT, the WT allows an analysis in both time and frequency domains giving information on the evolution of the frequency content of a signal over time.

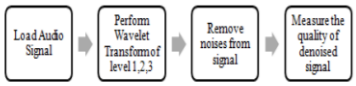


Figure 1 : Steps used for audio denoising.

4 Tables

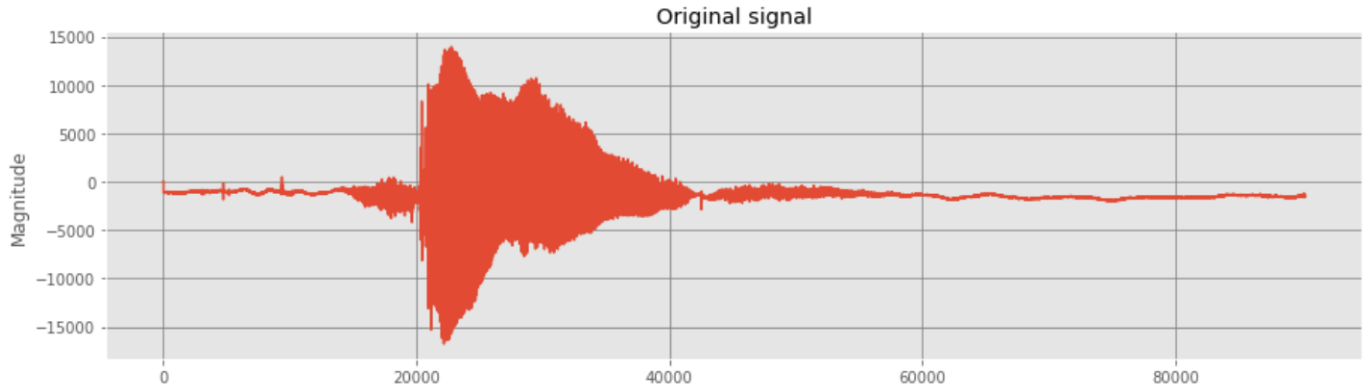
Table 1: SNR Comparison For White noise

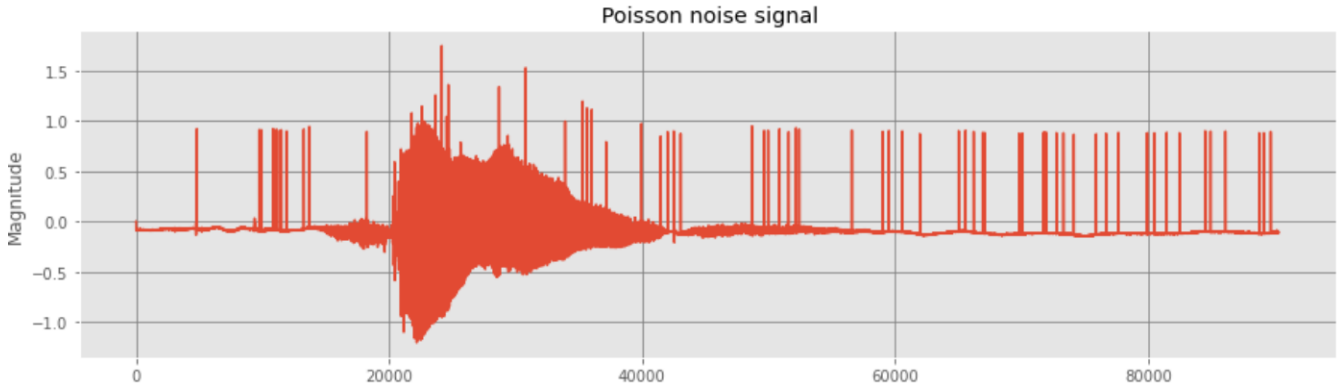
Method	SNR after Denoising
Wiener filter	114.43
Thresholding	14.78
HaarWavelet	15.824

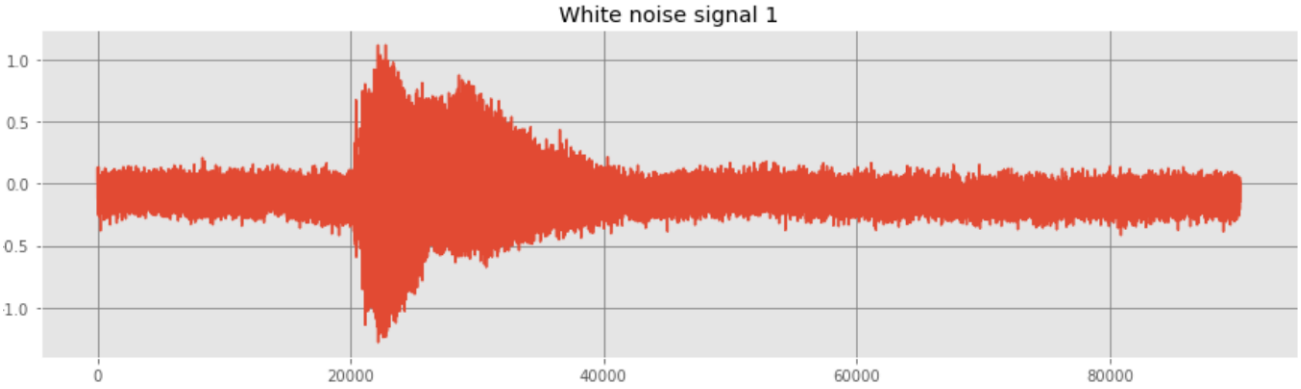
Table 2: SNR Comparison For Poisson noise

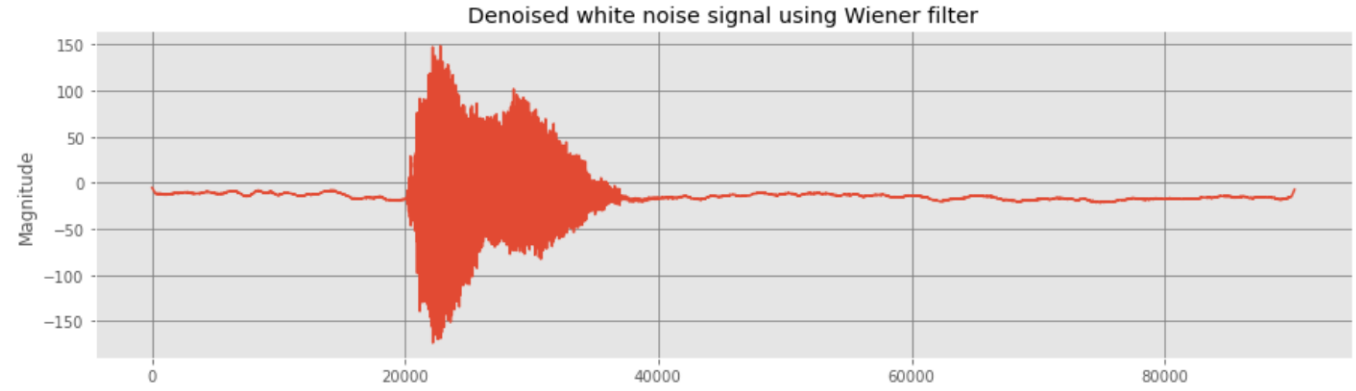
Method	SNR after Denoising
Wiener filter	130.87
Thresholding	25.97
HaarWavelet	33.31

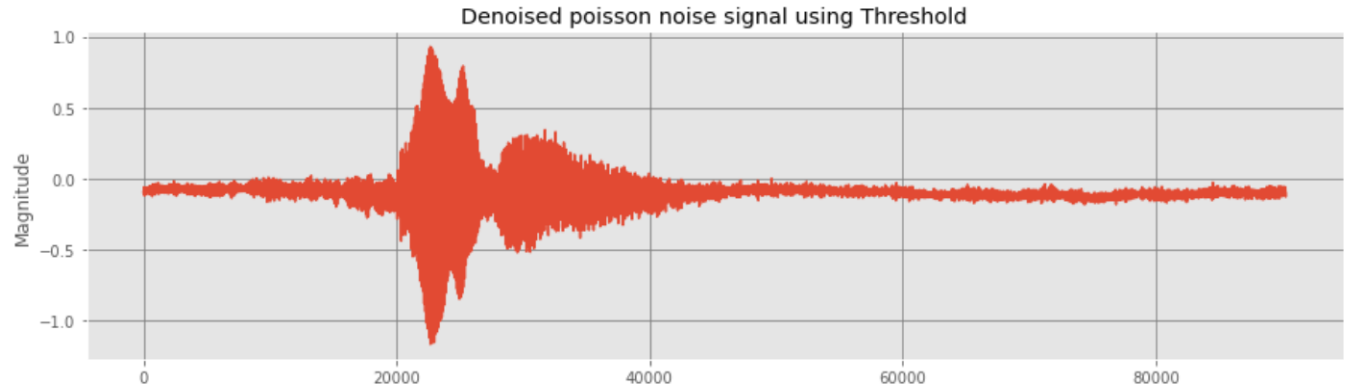
5 Output Images:

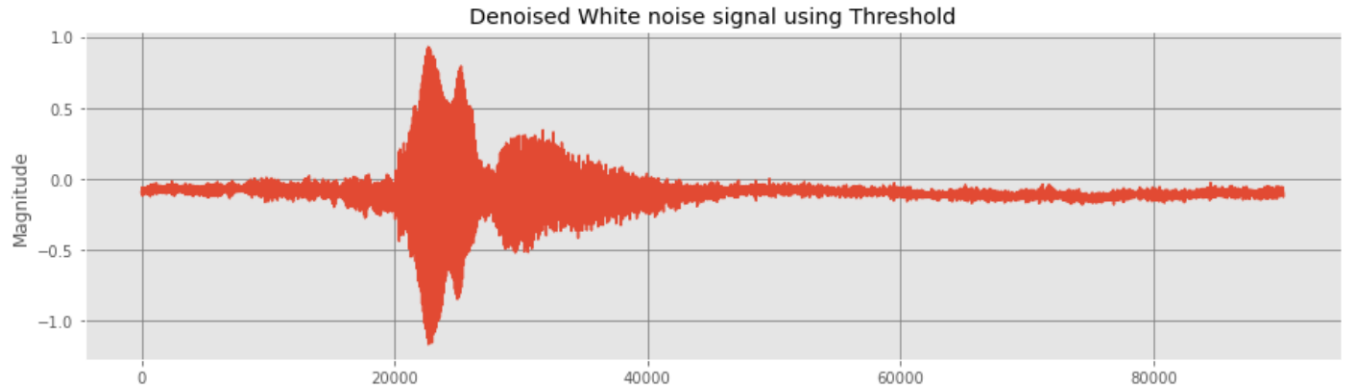












6 Conclusion

By doing these experiments we conclude that all fliter do not work perfect for same type of noise, selection of filter also depends on the type of the Noise.

7 References:

1) https://medium.com/swlh/noise-removal-for-a-better-fast-fourier-transformation-284918d4250f
2)https://nbviewer.jupyter.org/github/GuitarsAI/ADSP$_T$*utorials/blob/master/ADSP*1$2_W$*iener*$_F$*ilter.ipy*