260 Final Project

Matthew Dillabough
Zach Litzinger

November 2020

Overview

We created a simple wiener filter implementation which either takes noisy signals or generates a Gaussian white noise from un-noisy files and filters out the unwated noise. The implementation includes four primary functions:

- `generate_noisy_file()`
  - Simply applies a general Gaussian white noise sound to the signal for an un-noisy .wav file and the noisy signal over the original signal
- `wiener_filter()`
  - Given a .wav audio file, this function creates multiple frames of the audio signal
  - The function then iterates across all of these frames and calculates both the FFT and energy for each frame
  - These two values are then used to calculate the average energy per frame across the entire signal
  - The frames are then iterated over again (however, the size of the frame has been increased to overlap with the frames before and after it)
  - Using the FFT for this frame, we generate the magnitude and phase for the frame then calculate the signal to noise ratio (SNR)
  - For all data points within the frame, if the SNR value is above a set threshold (we hard-code the value to be 0.1), we reduce the value to the threshold
  - Finally, we adjust the magnitudes of the frequency bands to minimize the ones that are not desired in the filtered signal and then recombine all of the individually filtered frames to generate the filtered output
- `test()`

    - Given a .wav audio file (noisy signal), we plot the original signal, then apply our wiener filter and plot the two signals together to see what the filter has removed
    - Additionally, we take the output signal generated from

      ```
      wiener_filter()
      ```

      and generate a new clean .wav file for that signal

- `compare()`
    - Given a desired .wav file, this function simply plots the original audio signal (before the Gaussian white noise was applied) and the result of the wiener filter on said signal (with white noise applied) to compare the original file to our filter

Discussion

Purpose

This code quite obviously filters out unwanted noise from an audio signal. This can be very useful in cleaning signals which may have white noise or some extraneous sound which detracts from the desired sound of the signal. Applying our filter can help to remove some of this unwanted sound so the signal is clearer and easier to understand. While our filter does not entirely remove the white noise which we added to all of our test files, the results demonstrated in our compare function do prove that we were able to effectively remove much of the unwanted noise from these files. The filter is not perfect, but listening to the audio files proves that we were able to create clean files which are much clearer than the noisy inputs. Due to the simplicity of the algorithm, it runs quite efficiently, but we also did not test the filter on any sizeable audio signals (most of the speech snippets are no more than a few seconds long). Given that we both make music in our free time, this was a personally very interesting project and we will likely try to refine the algorithm so we can clean some the audio we use. For example, we do not use very high quality microphones to record our sounds so white noise can creep into the signals. It would be awesome to flesh out this algorithm to the point where we could actually use it in practice. One thing we found quite early on is that there are many ways you can fine tune these kinds of filters so there is a lot of room to play around with the parameters and structures to find out what works best.

Organization

We tried to keep the code very simple and so limited our implementation to a few function. All of the filtering takes place within the filter function and the other supplementary functions exist solely for testing purposes. Given that our implementation filters noise, we found it very useful for testing and to demonstrate the effectiveness of our code by visually displaying the signals at every step of the process (adding noise, filtering, etc.).

Challenges

The most significant challenge was certainly gathering audio files which fit into our implementation. Our implementation needs pretty specific files in order to function correctly (all 1600Hz sample rate, for example), so we needed audio signals which fit the constraints. If we were to expand upon this implementation we would like to do less hard-coding of the parameters for filtering these files to make it so our filter can handle all sorts of files with different properties. Another challenge was determining the values which resulted in the best filtering for the signals we used (i.e. our SNR threshold, alpha value, frame size, etc.). A lot of these values were finalized through a lot of trial and error so it would be interesting to make these values more dynamic to the specific signal that is being operated on. Finally, in the literature we researched prior to creating the implementation, there are many different opinions on what exactly is necessary to create the best, simple filter. These sources sometimes conflicted on how the signals should be operated on (i.e. whether or not to split the signal into frames and whether it is necessary to smooth the frames with a hanning function). We ended up trying out different implementations (using the different methods we found) and chose the one that gave us the best results for our test files.

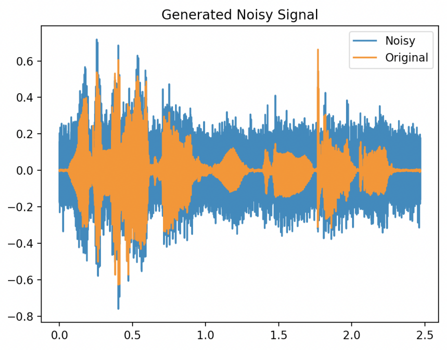
Test Files

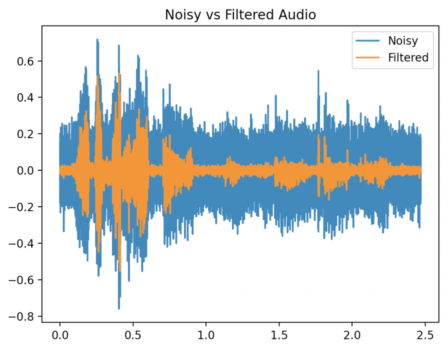
The code base contains three folders with all of our test files.

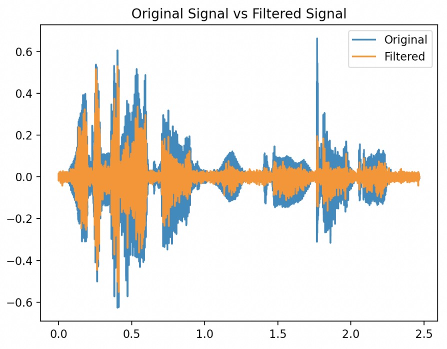
1. `test_audio`
    - 10 clean signals (.wav)
    - These signals are short speech snippets and very clear, they provide a solid benchmark of what the filtered signal (on the noisy version) should resemble
2. `noisy_audio`
    - Same 10 signals, but having been put through our noisy generator (.wav)
    - Given the clear test signals, these noisy audio files have a Gaussian white filter applied to them and will be filtered
3. `outputs`
    - The filtered signals on the 10 noisy signals (.wav)
    - These files are the output of wiener filter, we can compare these signals to the original, clean signals to test the effectiveness of our filter

Demonstration

The following screenshots show visualize the process of creating a noisy file, applying our wiener filter, then comparing the filtered signal to the original signal


Generated Noisy Signal
Noisy
Original


Noisy vs Filtered Audio
Noisy
Filtered


Original Signal vs Filtered Signal
Original
Filtered