INDEPENDENT STUDY REPORT
AUDIO DENOISING: A HETERODYNE APPROACH

Sreeganesh Valathara Rajendran
BU ID: U28069221

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May 2024

Technical Report No. ECE-2020-1

Summary

In this study, we present a two-stage audio-denoising system. In addition, we also present a novel method of audio-denoising that overcomes a fundamental problem with neural network training where MSE is involved. Our denoising pipeline involves two stages. The first stage is a variant of the conventional spectral-subtraction approach, borrowed from Sharma et al [1]. This approach involves estimating the noise spectrum and subtracting it from the input-spectrum to obtain the spectrum of the estimated clean-speech. The output of this method suppresses noise but in addition, introduces some artifacts. The second stage of the pipeline uses a data-driven neural network approach. This stage takes in both the noisy-frame as well as the spectral-subtracted frame to produce the clean-speech estimate. We introduce a novel method of denoising high-frequency components that reduces the infamous long-tail problem of neural networks which in turn also reducing training time.

Contents

1 Introduction **1**

2 Background **1**
2.1 Spectral Subtraction 2
2.2 Neural Networks 2
2.3 Frequency Shifting 3

3 Method **4**
3.1 Stage I: Spectral Subtraction Method 4
3.2 Stage II: Neural Network 6

4 Dataset **8**

5 Architecture **8**

6 Results **9**
6.1 Github Repository 10

List of Figures

1 MSE for Frame-To-Frame Denoising 9
2 MSE for Frame-To-DFT Denoising 9

1 Introduction

In audio, noise is any sound that is subjectively undesirable. In speech recordings, there is always a non-zero probability of noise entering the recording. In addition to being undesirable, the primary reason for removing noise is so it doesn't pose an obstruction to the task at hand. For example, in the task of Automatic Speech Transcription, the presence of noise results in the model producing the wrong transcription. In the case of interacting speech-based Artificial Assistants such as Google Assistant, Siri or Alexa, noise can result in the agent hearing the wrong input, which might result in it executing the wrong instruction execution.

Thus one of the primary obstacles facing voice-based technologies is the pervasive presence of noise in the audio streams recorded by microphones. Given the widespread adoption of speech-based technology, dedicated efforts towards denoising research and development are important. This need is further emphasized by the increasing prominence of edge devices as platforms for audio-based technologies. Since edge devices are extremely resource constrained, methods devised for it must be devised keeping efficiency and effectiveness in mind. In this study, we introduce a two-stage denoising pipeline designed to effectively reduce noise while maintaining the property of being computationally light.

Noise is of many types. The simplest division is additive noise and multiplicative noise. In this investigation, we're only dealing with additive noise as this is the most common form of noise encountered by speech-based technologies. A speech audio corrupted by additive noise is formulated in the following manner

$$y(t) = x(t) + n(t)$$

where $y(t)$ represents the noisy-speech in the time domain, $x(t)$ represents the clean-speech in the time domain and $n(t)$ represents the additive noise in the time-domain.

The pipeline we present in this study comprises of two approaches in series: spectral subtraction and neural networks. In stage I, we use spectral subtraction, which comprises of estimating the noise-spectrum from the noisy-speech and subtracting it from the input-spectrum to obtain the noise-suppressed clean-speech estimate. The frames from this stage are then passed onto stage-II. Stage-II of the pipeline takes in both the frames of the noisy-speech as well as that of the spectral-subtracted speech and feeds it into four neural networks, whose outputs are combined in a particular way to produce the clean-speech estimate.

2 Background

In this study, we borrow three different concepts to build our denoising system: spectral subtraction, neural networks and frequency-shifting. Spectral subtraction is one

of the classical methods in audio denoising that involves estimating noise spectrum and subtracting it from the input-spectrum to estimate the spectrum of the clean-speech. Neural networks are powerful function approximators that are widely deployed due to their ability and the rising ubiquity of systems with high computational power. And finally, we borrow a fundamental concept from communication theory called heterodyning, which we use to implement frequency shifting.

2.1 Spectral Subtraction

Spectral subtraction is a classical and conventional method used for noise-suppression. This method assumes that the noise is additive and is formulated in the following manner

$$y(t) = x(t) + n(t)$$
$$Y(e^{jw}) = X(e^{jw}) + N(e^{jw})$$

where y(t) is the noisy-speech, x(t) is the clean-speech and n(t) is the noise in the time domain. Similarly, $Y(e^{jw})$ is the noisy-speech, $X(e^{jw})$ is the clean speech and $N(e^{jw})$ is the noise in the frequency domain.

Once we estimate the noise spectra, we subtract it to obtain the speech-subtracted spectra in the following manner

$$\hat{X}(e^{jw}) = Y(e^{jw}) - \hat{N}(e^{jw})$$

, where $\hat{X}(e^{jw})$ is the estimated spectrum of clean-speech and $\hat{N}(e^{jw})$ is the estimated noise-spectra.

This is the fundamental approach of spectral subtraction. The different methods of spectral subtraction varies in the way that noise is estimated and the different ways in which the noise spectra is subtracted from that of the noisy-speech spectra. The reason we have different methods is fundamentally to avoid the problem of artifacts, which is pervasive wherever spectral subtraction is involved.

2.2 Neural Networks

Neural networks are a class of machine learning algorithms that is inspired by the structure and function of human and animal brains. They are powerful function approximators that consist of interconnected nodes, organized in layers. The input layer receives the data, which in our case, is a vector corresponding to the amplitudes at each sample, which is then processed through hidden layers using weighted connections. Each neuron in these layers apply a transformation to the input data based on its learned parameters, and the output layer produces the final result. Through backpropagation [**?**], the weights of each neurons at each layer are adjusted so as to

minimise a loss function which results in the neural network instantiating a function, whose input-output pairing is a super-set of the pairings present in the dataset.

Neural networks are often considered to be black-boxes due to their complex internal mechanisms and the "machine learning" aspect of the way they learn their weights. However, their power lies in their ability to learn from data without explicitly embedding information, in addition to their ability to learn mappings that are practically impossible or tedious with the current classical methods. This makes them incredibly versatile across various domains. In computer vision, neural networks have excelled at tasks like image classification, object detection and facial recognition. Recurrent Neural Networks, with their sequential processing capabilities, are well suited for natural language processing tasks such as language translation, sentiment analysis and speech recognition.

Neural networks have proven to be highly effective in audio applications, revolutionizing how sound is processed, analyzed and even generated [**?**]. Architectures such as classical fully connected networks (FCNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have shown promise in their ability to present competent performance in the context of audio. In this project, we mainly work with shallow FCNs due to their reduced computational load since we're designing this method keeping in mind compute and latency limitations.

From an implementation point-of-view, FCNs are essentially a number of sequential matrix multiplications with non-linearities applied in between multiplications. A 5 layered neural network, for example, is basically 5 matrix multiplications with non-linearities, such as ReLU or Sigmoid, applied element-wise after each multiplication. FCNs are thus desirable because matrix multiplications are such fundamental operations that every CPU, whether for edge or high-performance computing, is designed to highly optimize matrix multiplications and element-wise operations compared to other specialized operations such as convolutions. Thus, we stick to classical FCNs for this project and then move onto more sophisticated networks in future ventures.

2.3 Frequency Shifting

Given a signal, $x(t)$, we can shift its high-frequency components to its low-frequency components in the following manner, a fundamental property of fourier transform.

$$x(t) \leftrightarrow X(w)$$
$$x(t)e^{jw_0 t} \leftrightarrow X(w - w_0)$$

We perform this frequency-shift operation, given in Algorithm 1, for both our input data: the noisy-frame and the spectral-subtracted frame. This operation is conducted to better capture and denoise the high-frequency components of speech.

During deployment, however, instead of multiplying with the complex vector, we

instead multiply with a cosine of the same frequency with amplitude, 2. This is desirable because a cosine is a sum of e^{jw_0t} and e^{-jw_0t}. So when we multiply a signal with a cosine, this results in the frequencies of the "carrier" signal being shifted to the left and right by the frequency of the cosine. We then filter it accordingly to obtain the results we want.

Though this procedure and the procedure present in the Algorithm 1 essentially produce the same results, this is more computationally friendly because multiplication of two real signals contain less number of multiplications and additions compared to the multiplication between real and imaginary. Though the difference maybe negligible per operation, the difference is non-trivial because this is step that is performed for all the frames.

Algorithm 1 Frequency-shifting Audio

procedure FREQUENCY-SHIFT SIGNAL($x(t)$)
  $x_{\text{hp}}(t) \leftarrow$ HIGHPASSFILTER($x(t)$) ▷ High-pass filter
  $x_{\text{mod}}(t) \leftarrow x_{\text{hp}}(t) \cdot e^{jwt}$ ▷ Frequency-shift with e^{jwt}
  $x_{\text{lp}}(t) \leftarrow$ LOWPASSFILTER($x_{\text{mod}}(t)$) ▷ Low-pass filter
  return $x_{\text{lp}}(t)$
end procedure

3 Method

3.1 Stage I: Spectral Subtraction Method

Stage-I of our audio-denoising pipeline is a Multi-band spectral subtraction method presented by Sharma et al [1], called, "Multi Frame MultiBand Spectral Subtraction (MF-MBSS)". In the thesis, he presents a Voice-Inactivity-Detector (VID) that is used to detect frames that do not contain speech. These frames are used to estimate the noise-spectrum. The estimated noise-spectrum is subtracted from the input-spectra using MF-MBSS to obtain the spectral-subtracted data. This is the output of the first-stage of our denoising pipeline.

3.1.1 Noise Estimation

The VID [1] is used to find the frames that do not contain speech. Thus, these frames are the most likely to contain noise. Using the frames that contain speech, to estimate noise spectrum is not desirable because if we use those frames to subtract noise, the resulting spectra will also have speech suppressed due to the presence of speech spectra in the noise-estimate. The higher the content of speech-spectra in our noise-estimate, the higher speech will be suppressed in the resulting audio. Hence in this context, False Positive are more harmful than True Negatives. The VID is

designed keeping this in mind. The frames that do not contain speech are termed, "Definitely Speech Inactive" or DSI.

Once the DSI frames in the last 100ms are identified, the noise spectra is obtained by taking the average of its power, as shown below

$$\hat{N} = \frac{\sum_{i=0}^{M-1} |X_{DSI}|^2}{M}$$

3.1.2 Spectral Subtraction

Once the noise-spectra is estimated, we use it to subtract from the input-spectra. However, direct subtraction has shown to produce artifacts in previous approaches. To overcome this, we use the MF-MBSS method presented by Sharma et al [1]. In this method, the subtraction is engineered in such a way as to minimize the presence of artifacts in the output of the spectral-subtraction.

MF-MBSS performs spectral subtraction on overlapping frequency bands of frame sequences, obtained from STFT. Each frame-sequence is processed by computing the spectral subtraction parameters on frequency bands of 1000Hz with an overlap of 500Hz. The method consists of the following steps

1. Calculate Mixture-to-Noise-Ratio (MNR) for each frequency band in an audio-frame. The MNR is defined as follows

   $$MNR = 10log_{10}\{\frac{\sum |\hat{X}_i[k]|}{\sum |\hat{D}_i[k]|}\}$$

   , where $X_i[k]$ and $D_i[k]$ are the spectrum bin values at the i-th frequency-band.

2. The subtraction parameter, alpha is then obtained from the MNR using the following equation. This obtained alpha is then multiplied with another tuning factor, called, "aggressiveness parameter", which we empirically chose to have the value of 10.

   $$\alpha = \begin{cases} 5 - \frac{4}{25}(mnr + 5) & \text{for } -5 \leq mnr \leq 20 \\ 5 & \text{for } mnr \leq -5 \\ 1 & \text{for } mnr \geq 20 \end{cases}$$

3. The α values are then smoothed across frequency bands to reduce artifacts due to vanilla spectral subtraction.

4. The noise spectra is then multiplied with the α values and subtracted from the spectra of input-frame. The process is as follows

   $$\hat{Y}[k] = \begin{cases} Y[k] - \alpha\hat{N}[k] & \text{for } Y[k] \geq \alpha\hat{N}[k] \\ \beta Y[k] & \text{for } Y[k] < \alpha\hat{N}[k] \end{cases}$$

, where β is the floor spectral subtraction parameter assigned the value of 0.002.

5. The subtracted spectra is then smoothed across bands, frames and sequences. Please refer to Sharma et al [1] for more detailed steps.

The result of this set of processes gives us the spectral-subtracted STFT. This is then passed onto stage II of our audio-denoising pipeline.

3.2 Stage II: Neural Network

Stage-II of the audio-denoising system utilizes two classes of neural networks: magnitude networks and phase-networks. Magnitude networks are tasked with producing the magnitude of FFT given an audio-frame of sample length, 320, as input. Similarly, phase-networks are tasked with producing the phase of FFT given an audio-frame of sample length, 320, as input. To produce the clean-speech estimate, we break down the process of denoising into two parallel branches. The first branch takes care of denoising the low-frequency components, whose results we term, "frame A". The second branch is tasked with denoising the high-frequency components, whose results we term, "frame B". We use a separate magnitude-network and phase-network for each branch. Thus, in this stage, we use four neural networks in parallel, two for branch-A and two for branch-B. All the four neural networks take as inputs the noisy-audio frame and spectral-subtracted frame.

3.2.1 Denoising Low-Frequency Components

To denoise the low-frequency components, we low-pass filter both the noisy-frame and spectral-subtracted frame. These are fed to the magnitude-network of frame-A, $f_{A,magnitude} : \mathbb{R}^{320} \rightarrow \mathbb{R}^{257}$, and phase network of frame-A, $f_{A,phase} : \mathbb{R}^{320} \rightarrow \mathbb{R}^{257}$. The outputs of these are used to produce the 512-bin FFT.

3.2.2 Denoising High-Frequency Components

To denoise the high-frequency components, we high-pass filter the noisy-frame and spectral-subtracted frame with filter cut-off at 4000 Hz. These are then frequency shifted to baseband such that the frequency components at 4000 Hz to 8000 Hz now exist at 0 Hz to 4000 Hz. The result of this step is low-pass filtered with cut-off frequency = 4000 Hz to remove any potential aliasing. These are fed to the magnitude-network of frame-B, $f_{B,magnitude} : \mathbb{R}^{320} \rightarrow \mathbb{R}^{257}$, and phase network of frame-B, $f_{B,phase} : \mathbb{R}^{320} \rightarrow \mathbb{R}^{257}$. The outputs of these are used to produce the 512-bin FFT. The result is low-pass filtered with cut-off frequency = 4000 Hz, because there is no guarantee that the outputs produced by this process is bandlimited to 0 - 4000 Hz. So not low-pass filtering will result in aliasing when frequency shifting. The result of this band-limiting step is then frequency-shifted back to its original band.

After this step, the signal will have its bins have non-zero values at 4000 Hz - 8000 Hz. The result is then low-pass filtered again to remove any potential aliasing.

Note that the outputs of the network are only for bins from 0 to π since the FFT of a real-sequence is conjugate symmetric. The magnitude and phase values for bins beyond the Nyquist frequency (257th bin) are symmetrically related to those before it. By mirroring the magnitude values and negating the phase values, we can reconstruct the FFT. This reduction in dimensionality of neural network allows us to achieve our task while simultaneously reducing compute complexity and latency. In addition, it is not a good idea to estimate the whole spectrum using a neural network because any slight change can result in the failure of the conjugate symmetry property, thereby resulting in the IDFT not producing real signals.

3.2.3 Producing Final De-noised Outputs

Finally, Frame-A and Frame-B are added together. Since Frame-A is the denoised low-frequency components and Frame-B is the denoised high-frequency compoents, the sum of the two is basically the denoised input audio-frame.

Using the same approach, we also present the results when the neural networks take in a temporal-frame and also produce a temporal-frame. The two approaches both utilize heterodyning and provide an alternative approach to the same task at hand.

3.2.4 Why Two Sets of Magnitude and Phase Networks

Ideally, just training a magnitude-network and phase-network is enough since a 512-bin is enough to uniquely identify a 320 sample audio-frame. However, during our investigation in this direction, we found that the network consistently failed to reproduce the de-noised high-frequency components.

The conventional way around this is by using deeper networks, increasing size of dataset, and increasing number of epochs. However, using deeper networks are not preferred since this method is designed keeping deployment to edge devices in mind, where compute and energy is extremely constrained. In addition, if this method is to be inserted into a real-time pipeline, then additional depth in the layers will result in increased latency, which is extremely not preferred in real-time systems. In addition, training for more time produces reducing returns, referred to as the long-tail problem in machine learning.

It is to overcome these issues, that we use the proposed frequency-shifting approach. During our investigations, it was observed that when we train an audio-denoising network with MSE as the loss function, the network focussed on denoising the low-frequency components more than the high-frequency components. This is due to the fact that denoising low-frequency components reduced the MSE loss the most. That is, the MSE-loss reduced by denoising the low-frequency components

were significantly higher than that of the high-frequency components due to the fact that the amplitudes of low-frequency components were significantly higher than high-frequency components in an audio-signal, which translates to them being paid more attention by the neural network.

In addition, once the neural networks learned to successfully denoise the low-frequency components, the errors reduced significantly. When the error is extremely low, due to the nature of backpropagation, the gradient diminishes. This is one of the factors that contributes to the long-tail problem. When the gradient diminishes, the amount by which the weights are upgraded also reduces since the delta values is given by

$$\Delta = \nabla_{\text{gradient}} \alpha_{\text{learning rate}}$$

Thus, when working in audio-denoising, the neural networks prioritizes the low-frequency components over the high-frequency components. Our investigations showed that this resulted in the audio losing its, "crispness", since the high-frequency components are responsible for the crispiness of audio. Our method allows us to de-noise the high-frequency components better, which results in the entire frequency spectrum being de-noised.

4 Dataset

We use the TIMIT database throughout this project. TIMIT Acoustic-Phonetic Continuous Speech Corpus was a joint effort among the Massachusetts Institute of Technology (MIT), SRI International (SRI) and Texas Instruments, Inc. (TI). It comprises of approximately five hours of English speech along with time-aligned transcriptions. TIMIT contains broadband recordings of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences [2].

The TIMIT corpus includes time-aligned orthographic, phonetic and word transcriptions as well as a single channel, 16-bit, 16kHz speech waveform file for each utterance. Speaker metadata includes gender, dialect, birth date, height, race, and education level. Of the 630 speakers, about 70% are men and 30% are women[2].

5 Architecture

The Frame-To-Frame neural network architecture is a simple FCN with symmetric skip connections. The first and last layer has a length of 320 while the hidden layers have a length of 256. The skip connections are provided to improve the flow of gradients during training.

The Frame-To-DFT neural networks, both magnitude and phase networks, have the first layers contain 320 samples. The last layer contains 257 samples. There are

three hidden layers with the number of neurons being 257. Please refer to github code for better understanding of architecture.

6 Results

Here, we present the MSE distance of the two approaches: frame-to-frame denoising and frame-to-DFT denoising.

SNR	MSE, Input-Audio and Spectral Subtracted Audio	MSE, Input-Audio and Neural Network Audio
20	0.06421766885	0.07488805685
15	0.0914777422	0.08786381063
10	0.08774242697	0.08762384371
5	0.09660563275	0.09171097274
0	0.1149133499	0.1158810498
-5	0.1384355243	0.1439783427
-10	0.1706155951	0.1917237484
-15	0.1403897777	0.1772681848
-20	0.1599107869	0.2833204798

Figure 1: MSE for Frame-To-Frame Denoising

SNR	MSE, Input-Audio and Spectral Subtracted Audio	MSE, Input-Audio and Neural Network Audio
10	0.09385113487	0.07472554439
5	0.09588582147	0.07824850691
0	0.0858276432	0.09676622928
-5	0.08478126503	0.1270253923
-10	0.049317454	0.1811999846
-15	0.04309864215	0.2250842012
-20	0.0256667659	0.2363473243

Figure 2: MSE for Frame-To-DFT Denoising

From the low Mean Squared Error (MSE) values obtained, it's evident that among the two approaches explored, the Frame-to-Frame approach incorporating heterodyning stands out as the method yielding the lowest errors. However, these methods were designed to improves results of Google ASR. However, it was found that Google ASR refused to produce results when the SNR dropped below 0. This is presumably due to Google ASR's sensitivity to artifacts, since they are not encountered, in natural audio, which is used to train Google ASR.

However, while MSE serves as a metric for error calculation, it doesn't fully encapsulate the perceptual characteristics perceived by the human ear and brain. In

essence, a lower MSE doesn't guarantee that two audio clips will sound identical. This discrepancy between objective error metrics and subjective perception extends across various domains like audio, video, and images. Consequently, future research work must explore alternative loss functions, such as adversarial losses, which aim to denoise input audio in a manner that the listener perceives as having reduced noise.

Hence, in future research, it is important to merge the benefits of frame-to-frame approaches with perceptual losses. This integrated approach holds the promise of developing denoising systems that effectively reduce noise while also preserving the perceptual characteristics crucial for maintaining audio quality.

In summary, our study introduces a method for denoising audio using shallow neural networks. This methodology can also work with deep neural networks, offering the potential for enhanced performance. The advantages of this approach include its attaining high performance with less compute, rendering the task computationally and latency-friendly. Additionally, the method's inherent nature allows for reduced training times compared to standard methodologies. Moreover, employing multiple parallel shallow networks instead of a single deep network further minimizes pipeline latency, which makes it an excellent choice for tasks where the pipeline latency must be minimized as possible.

6.1 Github Repository

Link to github project page `https://github.com/vrsreeganesh/AudioDenoising_A_HeteroDyned_Approach.git`

References

[1] Sharma, Siddhant. "Voice Inactivity Ranking for Enhancement of Speech on Microphone Arrays." PhD thesis, Boston University, 2022. OpenBU, https://open.bu.edu/handle/2144/43714.

[2] Garofolo, John S., et al. TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium, 1993.