Speech Enhancement Techniques

ECE 6255 Final Project Report

Jie Deng
jdeng64@gatech.edu
Georgia Institute of Technology

Shan Jiang
sjiang340@gatech.edu
Georgia Institute of Technology

May 2, 2023

Abstract

This is the report for our final project - **Speech Enhancement Techniques**, for **ECE 6255**- *Digital Processing of Speech Signal* at the Georgia Institute of Technology, Atlanta, Georgia, Spring 2023. In this report, we started by introducing the background and motivation of our project in the introduction section. Then, we formally introduced the speech enhancement problem and introduced the methods and approaches we used for our project. After that, we demonstrated the dataset we used, our experiments setup, the evaluation metrics, and analyzed our results and findings. Finally, we drew our conclusion, explained the difficulties we had, and discussed potential future improvements and study directions.

1 Introduction

In digital signal processing (DSP), speech enhancement refers to the task of improving speech quality. The objective of enhancement is improvement in intelligibility and overall perceptual quality of degraded speech signal DSP techniques. It has a wide range of applications including mobile phones, teleconferencing systems, speech recognition, speaker diarization, hearing aids, etc. It has been a hot topic for years and is still an active area of research. In our project, we explored speech enhancement techniques including traditional DSP methods and deep learning-based approaches, with a focus on speech denoising. We applied what we have learned this semester in ECE 6255 to analyze these methods and compare their performance from various aspects. During this process, we not only got a better understanding of speech enhancement but also reinforced concepts that we have learned in lectures.

In this paper, we first reviewed the related works for speech denoising tasks and briefly introduced the normal approaches, such as the rule-based algorithm Wiener Filter, or the first DNN model for speech denoising proposed in 2014[1]. By then, we discussed the basic concepts and the implementation principles of the 4 methods we want to compare, including Wiener Filter in Frequency Domain, LogMMSE, NSNET2, and DeepFilterNetwork. The choices of these four methods are based on the assumption that:

- The Wiener Filter is a good baseline system.
- LogMMSE properly represents the SOTA performance of the traditional algorithms.
- NSNET2 is a relatively simple DNN model.
- DeepFilterNetwork is more complex and has higher performance.

Thus, these four methods are representative of both traditional DSP denoising methods and modern DNN models.

After that, in the Experimental Results and Analysis section, we introduced the dataset and the implementation details, as well as the final result analysis. By examining the results, we concluded that DNN is largely supreme to the traditional rule-based algorithm. Compared with the Wiener Filter or LogMMSE, the DNN is highly good at recognizing the noise magnitude distribution pattern, and reducing them, especially in the non-stationary noise cases. However, both the training and predicting phase of DNN models consume a huge amount of computational power, which limited its usage in mobile devices, where the power consumption is strictly regulated by its size and mobility.

2 Related Work

For the denoising problem, there were a huge amount of previously researched methods that achieved great levels in terms of both intelligibility and quality.

Prior to the emergence of Deep Neural Networks, the rule-based algorithm families can be classified into two categories, one is the Multiple Sensor Methods, and the other is the Single Sensor Methods. The Multiple Sensor Methods normally include Adaptive Noise Cancellation and Beam-forming Noise Suppression, and the Single Sensor Methods include Wiener Filter, Kalman Filter, Spectral Estimation (MMSE, MAP), Restoration via Model-based Analysis Synthesis, Spectral Subtraction, Wavelet Denoising, etc.

Specifically, the Multiple Sensor Method takes more than one microphone as the input signal. In Adaptive Noise Cancellation applications, one extra signal channel is often used to capture the reference noise, such that to dynamically analyze the statistical possibility model of the noise, and to adapt the filter coefficients to suppress the noise. In Beam-forming Noise Suppression applications, the noise and signal are located at different positions, by utilizing the near-field beam-forming algorithm, the noise located at the off-axis positions will be suppressed.

Among the Single Sensor Methods, the most popular algorithm family is the Wiener Filter, it estimates the convolutional and additive noise by multiplying the auto-correlation and cross-correlation between the received signal and the training sequence. It can be expressed as below, where R_y^{-1} represents the inverse of the auto-correlation of the received signal and r_{dy} represents the cross-correlation of the training and received signal.

$$w = R_y^{-1} * r_{dy}$$

However, calculating the inversion of the correlation matrix in the Wiener-Hopf equations is not very practical in the real-time speech processing system. There are several algorithms targeting approaching the optimized estimation of the system function. LMS was proposed by Bernard Widrow and Ted Hoff in 1960 [26], it mimics the unknown system function using the stochastic gradient descent method.

Besides the Wiener Filter, there are a variety of algorithms based on traditional statistical or mathematical methods to estimate and reduce the noise in speech. An early model is Spectral Subtraction [39], which manually estimates and deducts the noise based on DFTA (Discrete Fourier Transform Analysis). Another well-known method is Wavelet Denoising, firstly proposed by Johnstone and Silverman in 1997[40].

More recently, as the Deep Neural Network method for speech enhancement proposed by Chin-Hui, Lee[1] the overall performance has been significantly improved. Thus, more researchers are starting to look at the Deep Learning field.

3 Problem formulation

As shown in our introduction and literature review sections, lots of approaches have been developed and applied to the task of speech denoising during the past few decades. On a high level, these techniques can be split into two categories: (1) Conventional DSP techniques and (2) DL Techniques. For the purpose of this project, our goal is to explore as many techniques as we can and compare their performance using various evaluation metrics. To achieve this goal, we selected two methods from the DSP category and two methods from the DL category. In the following section, we are going to explain them in detail.

3.1 Conventional DSP Techniques

3.1.1 Wiener Filter

The Wiener Filter is being widely used as a popular denoising method, and it has a number of variations in different scenarios. In this experiment, we utilize the Wiener Filter in the frequency domain method.

A typical Wiener Filter block diagram is shown below. In this case, the desired signal is being transmitted by the system, thus causing signal loss or noise addition. To recover the desired signal d from the received signal y, we need to create a wiener filter with the inverse system response h as opposed to the system distortion hx.

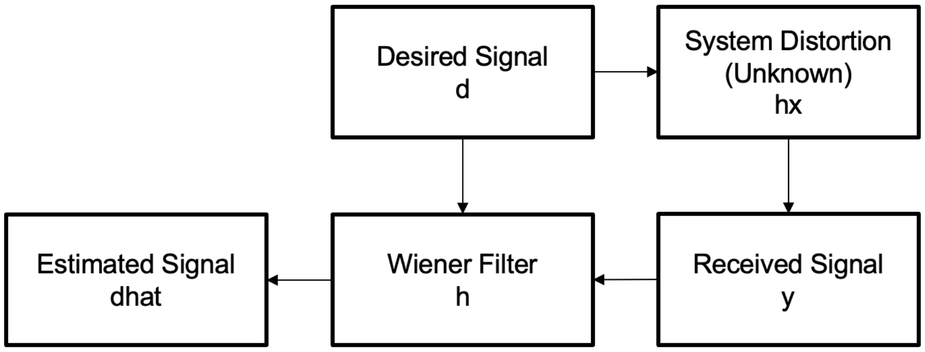

Figure 1: A typical structure of Wiener Filter

To get the coefficients of the Wiener Filter, we take the gradient of the error and let it equal zero to find the minimum error. The derivation steps are shown below:

Firstly, the error can be written in this form 2:

$$\hat{d}(n) = \sum_{k=-\infty}^{\infty} h_k y(n-k), -\infty < n < \infty \tag{1}$$

$$error[n] = d[n] - \hat{d}[n] \tag{2}$$

Do the DTFT for the error 3:

$$Err(\omega_k) = D(\omega_k) - \hat{D}(\omega_k) = D(\omega_k) - H(\omega_k)Y(\omega_k) \tag{3}$$

Take the gradient from the square of the error 5:

$$\varepsilon[|Err(\omega_k)|^2] = E[[D(\omega_k) - H(\omega_k)Y(\omega_k)]^*[D(\omega_k) - H(\omega_k)Y(\omega_k)]] \tag{4}$$

$$\frac{\partial(|Err(\omega_k)|^2)}{\partial H(\omega_k)} = 0 \tag{5}$$

Rearrange the equation, get 7:

$$H(\omega_k) = \frac{P_{dy}(\omega_k)}{P_{yy}(\omega_k)} \tag{6}$$

$$H[k] = \frac{P_{dy}}{P_{yy}} = \frac{P_{yy} - P_{nn}}{P_{yy}} = 1 - \frac{P_{nn}}{P_{yy}} \tag{7}$$

3.1.2 LogMMSE

Among the conventional rule-based denoising algorithm, MMSE has a relatively high noise suppression level dealing with various noise types.

A simple diagram is given below. In this case, the MMSE estimator predicts the next $Xhat$ by taking the a posterior and a priori SNR as a factor to multiply $Y[k,n]$.

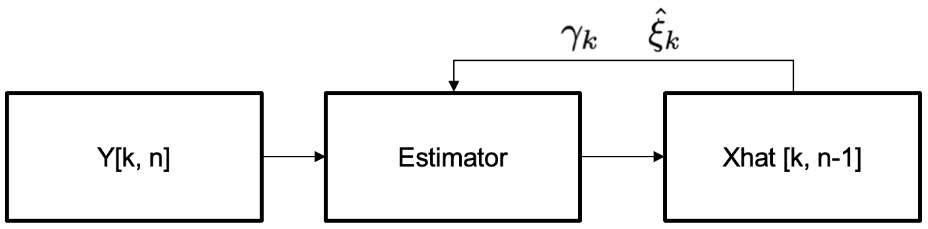

Figure 2: A typical structure of MMSE

The intuition behind this is, if we know the ratio between the noise power and the signal power in the noisy signal, we can estimate the magnitude of the signal based on the noisy signal, which can be described by 8:

$$\hat{X}_k = f(\frac{\lambda_y - \lambda_d}{\lambda_y}) * Y_k \tag{8}$$

The detailed steps of MMSE are given below :

$$Y_k = DFT[y[n]] \tag{9}$$

$$\lambda_d(k) = DFT[noise^2[n]] \tag{10}$$

$$\gamma_k = \frac{Y_k^2}{\lambda_d(k)} \tag{11}$$

$$\hat{\xi}_k = \alpha \frac{\hat{X}_k^2(m-1)}{\lambda_d(k,m-1)} + (1-\alpha)max[\gamma_k(m)-1,0] \tag{12}$$

$$v_k = \frac{\xi_k}{1+\xi}\gamma_k \tag{13}$$

$$\hat{X}_k = \frac{\sqrt{\pi}}{2}\frac{\sqrt{v_k}}{\gamma_k}\exp(-\frac{v_k}{2})[(1+v_k)I_0\frac{v_k}{2} + v_k I_1(\frac{v_k}{2})]Y_k \tag{14}$$

$$\hat{x}(n) = IDFT[\hat{X}(\omega_k)] \tag{15}$$

In this case, we use LogMMSE, which is replacing 14 with the equation below:

$$\hat{X}_k = \frac{\xi_k}{\xi_k+1}\exp(\frac{1}{2}\int_{v_k}^{\infty}\frac{e^{-t}}{t}dt)Y_k \tag{16}$$

3.2 Deep Learning Techniques

3.2.1 NSNET-2

The first deep learning method we selected is using the idea of masking. The Noise Suppression Net(NSNET-2) was based on ideas proposed in [30] by Sebastian Braun and Ivan Tashev at Microsft Research in 2020. In pure noise reduction task, the authors assumed that the observed signal is an additive of the desired speech and noise, and the mathematical formulation can be written as follows: 17.

$$X(k, n) = S(k, n) + N(k, n) \tag{17}$$

- k: frequency index.
- n: time index.
- $X(k, n)$: the Short-Time-Fourier Transform(STFT) of the mixed signal.
- $S(k, n)$ the STFT of the clean speech signal.
- $N(k, n)$ is the STFT of the additive noise.

The goal is to construct a filter $G(k, n)$ in the STFT domain to recover the STFT of the clean signal by applying it to the observed signal, as shown in 18.

$$\hat{S}(k, n) = G(k, n)X(k, n) \tag{18}$$

In practice, this mask can be a real-values suppression gain, or a complex-valued filter. While the former option (also known as mask) only recovers the speech amplitude, while a complex-valued filter could also potentially recover the signal phase. In this paper, the authors used the former option.

The Network been used is shown in figure 3.2.1.

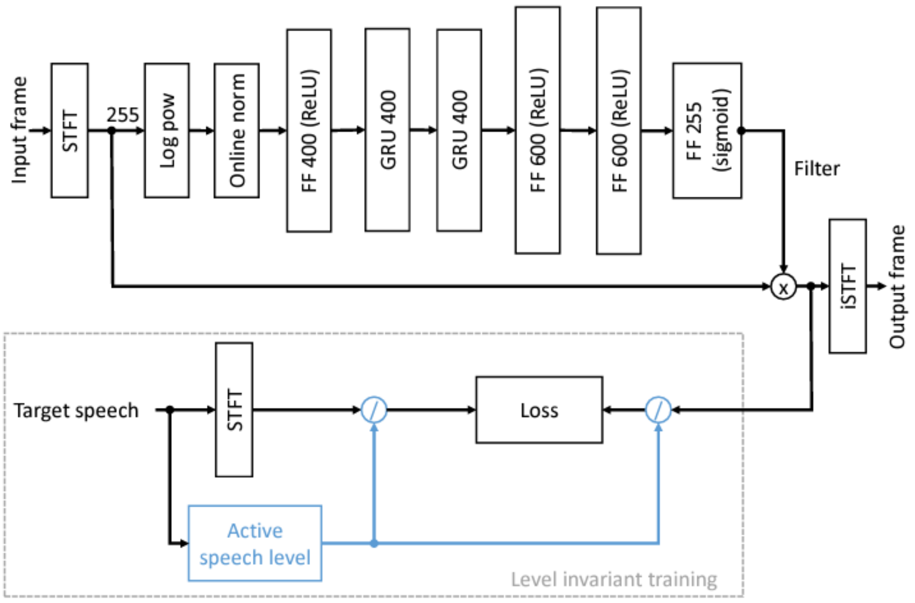

Figure 3: Overall structure of NSNET2 [35]

This recurrent network architecture is based on Gated Recurrent Units(GRUs)[31] and feed forward layers(FFs). Given an input frame, they first take the STFT using a 32ms square-root Hann Window and a shift of 16ms. They only kept the 1 to 255 frequency bins and dropped the first and 256 to 512 frequency bins because these do not carry useful information.

Then they take the log power spectrum(LPC) $P = log_{10}(|X(k,n)|^2 + \epsilon)$, and normalized the results by the global mean and variance of the training set. After that, the result is fed into an FF layer with Rectified Linear Unit(ReLU) activation, followed by two GRU layers, followed by another two FF layers with ReLU activation, and a last FF layer with sigmoid activation to ensure the output is positive. At this point, the mask $G(k,n)$ is generated and would be used to filter the original speech signal $X(k,n)$, the final clean speech would be recovered with ISTFT.

The training loss of the network is explained in the bottom half of figure . Formally in 19.

$$\mathcal{L} = \alpha \sum_{k,n} \left| |S|^c e^{j\phi S} - |\hat{S}|^c e^{j\phi \hat{S}} \right|^2 + (1-\alpha) \sum_{k,n} \left| |S|^c - |\hat{S}|^c \right|^2 \tag{19}$$

This level invariant normalized loss proposed by [33] and [34], and it captures the distance between the enhanced and original speech's spectral representations. The first half is based on both magnitude and phase while the second half only depends on magnitude. The choice of c and α are both 0.3.

For more neural network details, including the training epochs, learning rate,layer dimensions, please refer to the original paper [30] and their open source code [27].

The original paper also discussed data augmentation techniques to improve the performance of speech enhancement. For our project, we did not dive into that section and perform tests they did given the fact we only had limited time and computing resources.

3.2.2 DeepFilterNet

The second deep learning method we chose is using the idea of deep filtering. The Deep Filter Net (DFN) [35] was propose by Hendrik S. and Alberto N. in 2022. The DFN has a different mathematical formulation in comparison to NSNET-2, shown in 20

$$x(t) = s(t) * h(t) + z(t) \tag{20}$$

Here, $x(t)$ is the mixed noisy signal, $s(t)$ is the clean signal, $h(t)$ is the environment impulse response, and $z(t)$ is an additive noise. In the STFT domain 21:

$$X(k,f) = S(k,f) * H(k,f) + Z(k,f) \tag{21}$$

The deep filtering is defined by a complex filter in the STFT domain 22:

$$Y(k,f) = \sum_{i=0}^{K} C(K,i,f) \cdot X(k-i+l,f). \tag{22}$$

C is the complex coefficient of filter order N that are applied to the input spectrogram X. l is the look ahead which can take non-causal taps in the linear combination into count if $l \geq 1$. The authors also introduced a learned weight factor α to produce the final spectrogram to make sure deep filtering only affects periodic parts 23.

$$Y^{DF}(k,f) = \alpha(k) \cdot Y^{DF'}(k,f) + (1-\alpha(k)) \cdot Y^{G}(k,f). \tag{23}$$

Where Y^G is the gain-enhanced spectrogram.

The overall algorithm structure of DFN is shown in figure 4. As shown in the figure, on high-level, DFN can be summarized in two stages. The first stage predicts real-values gains G and enhances the speech envelope, giving result Y^G, the short time spectrum. The second stage predicts deep filtering coefficients C of order N, then these coefficients are applied to Y^G.

The noisy signal $x(t)$ is first transformed to the STFT domain, using a 30ms window and 50% overlap. The two types of features are calculated: (1) Equivalent rectangular bandwidth (ERB) features and (2) Complex features. The ERB features $X_{ERB}(k,b), b \in [0, N_{ERB}]$ are computed using log power spectrogram and normalized using exponential mean normalization [36], and applied rectangular

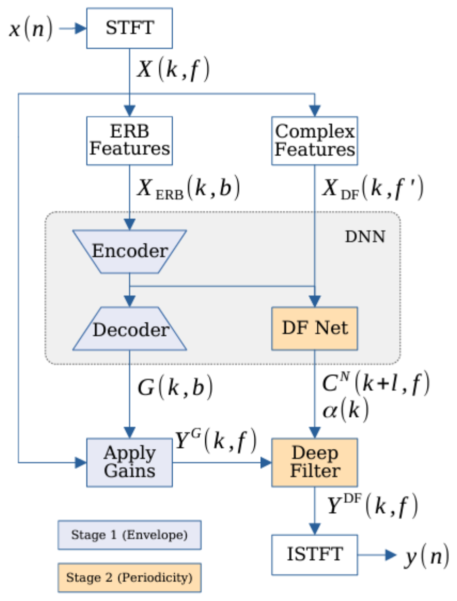

Figure 4: Overall structure of DFN [35]

ERB filter bank with N_{ERB} bands. The complex features $X_{DF}(k,f'), f \in [0, f_{DF}]$ is just the complex spectrogram with exponential unit normalization [37].

The encoder/decoder architecture is used to predict the ERB scaled gains $G(k,b)$. To transform the gains back to frequency domain, an inverse filter bank is applied. After that, point-wise multiplication with the noisy spectrogram is performed.

The DF Net architecture is shown in 5. It is composed of simple DNN layers including convolution, batch normalization, ReLU, etc. The authors also utilized a U-Net [38] like structure when building the whole network. For more details on dimension of each layer, please refer to the original paper.

The loss of for DFN training is composed of two parts, first, 24 defines the compress spectral loss, which is very similar to what we have seen in NSNET2.

$$\mathcal{L} = \sum_{k,f} \left| |Y|^c e^{j\phi Y} - |S|^c e^{j\phi S} \right|^2 + \sum_{k,f} \left| |Y|^c - |S|^c \right|^2 \tag{24}$$

The second component of the loss is added to force the DF component to only enhance periodic parts of the signal because DF can not provide any benefit over ERB gains for noise-only sections and it may even cause artifacts by modeling periodic noises like engine or babble noise which is most noticeable in attenuation limited models, and DF can not provide any benefits to random components like fricatives or plosives. Under the assumption that these sections have most energy in higher frequencies, the authors computes the local SNR for frequencies below f_{DF}, and defined this loss to be 25:

$$\mathcal{L}_\alpha = \sum_k ||\alpha \cdot \mathbb{1}_{LSNR<-10dB}||^2 + \sum_k ||(1-\alpha) \cdot \mathbb{1}_{LSNR>5dB}|| \tag{25}$$

where $LSNR$ stands for the local SNR and $\mathbb{1}$ is the indicator function. The total loss is given by 26:

$$\mathcal{L} = \lambda_{spec}\mathcal{L}_{spec} + \lambda_\alpha \mathcal{L}_\alpha \tag{26}$$

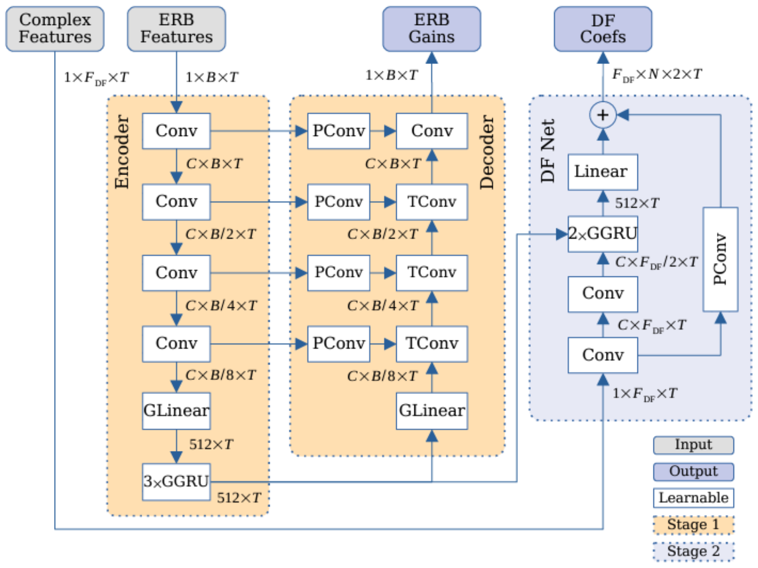

Figure 5: Overall structure of DFN
[35]

4 Experimental results and analysis

4.1 Dataset Preparation

The main dataset we are using is the Noisy speech database for training speech enhancement algorithms and TTS(Text-to-Speech) models [21]. It is a clean and noisy parallel speech database and it is designed to train and test speech enhancement techniques that operates at 48kHz. It contains 28 speakers - 14 male and 14 female of the same accent region (England) and another 56 speakers - 28 male and 28 female of different accent regions (Scotland and United States). There are around 400 sentences available from each speaker. All data is sampled at 48kHz and orthographic transcription is also available [22]. All speakers in the dataset are selected from the Voice Bank Corpus [23]. Some of the noises used to create the noisy speech were obtained from the Demand database [24]. Other speech-shaped and babble noise files are from [25]. The speech-shaped noise was created by filtering white noise with a filter whose frequency response matched that of the long term speech level of a male speaker. The babble noise was generated by adding speech from six speakers from the Voice Band Corpus that were not used for either training or testing. The other eight noises were selected using the first channel of the 48kHZ versions of the noise recordings of the Demand database. The noises are from 8 scenarios: a domesitic noise (inside a kitchen), an office noise (in a meeting room), three public space noises (cafeteria, restaurant, subway station), two transportation noises (car and metro), and a street noise (busy traffic intersection) [22].

We are choosing this dataset because (1) It has been widely used by researchers for training and testing speech enhancement and TTS methods.(2) It contains various types of noises. In lectures, we learned that classical DSP techniques might only perform well in filtering certain types stationary noises, but since our goal is to compare their performance with deep learning methods in different noisy scenarios, it is important for us to ensure that our dataset contains various types of non-stationary noises.

It is also important to notice that, the original dataset didn't separate noisy signals at different signal-to-noise ratio (SNR), but we are also interested in comparing the performance of denoising methods when noises and signals are mixed using different SNRs, therefore, we took the original dataset and aumengted it to create noisy signals at different SNR levels. First, we take the original noisy signal and subtracted it by its clean version, then, we remixed the clean version and the noise at these SNR levels: -5 dB, 0 dB, 5 dB, 7 dB, 10 dB, 15 dB, 20 dB. All these SNR levels have appeared in the training dataset, but except the 7 dB, we intentionally added this inspired by [1]. We added this as a mismatching test because we want to test if the models are still going to perform well for SNRs that has never appeared in the training set.

4.2 Implementation and Codes

Our Wiener Filter and LogMMSE methods are implemented in MATLAB with reference to [29].

The NSNET-2 and DeepFilterNet are implemented in Python using PyTorch. It is important to note that for the deep learning methods, we didn't perform our own implementation but instead used the GitHub open source code at [27] and [28] respectively. Since implementation is not the major focus of this project, we investigated more time in studying the concepts and analyzing the performance of our models.

Some of the core code implemented by us are provided in the Appendix section, and links to other open source code we used or referred are also provided there.

4.3 Evaluation Metrics

To evaluate the performance of our methods, we mainly referred to [1] [2] [3] [4] [5]. We eventually chose SNR and PESQ as our evaluation mectrics.

4.3.1 Signal-to-Noise Ratio

The general form of SNR can be written as below, it describes the ratio between the power of signal and the power of noise:

$$SNR = \frac{RMS_x}{RMS_{noise}} = 10\log 10 \frac{\sum_{n=1}^{N_0} x^2[n]}{\sum_{n=1}^{N_0} noise^2[n]} = 10\log 10 \frac{\sum_{n=1}^{N_0} x^2[n]}{\sum_{n=1}^{N_0} (x[n]-\hat{x}[n])^2} \tag{27}$$

Speech signal is non-stationary, its corresponding SNR value is also varying. To best represent the SNR within a short period of time, we need the local SNR as opposed to the global SNR, which can be defined as below:

$$localSNR = 10\log 10 \frac{\sum_{n=Nm}^{Nm+N-1} x^2[n]}{\sum_{n=Nm}^{Nm+N-1} (x[n]-\hat{x}[n])^2} \tag{28}$$

Moreover, we want a method that not only representing the variation of SNR between blocks, but also representing the overall SNR performance. By adding the local SNR together then average them, we get the segmental SNR:

$$segSNR = \frac{10}{M}\sum_{m=0}^{M-1} \log 10 \frac{\sum_{n=Nm}^{Nm+N-1} x^2[n]}{\sum_{n=Nm}^{Nm+N-1} (x[n]-\hat{x}[n])^2} \tag{29}$$

4.3.2 Perceptual evaluation of speech quality (PESQ)

This speech quality metric is being widely used for Noise Reduction Systems. Rather than the previous bark spectral distortion (BSD), the perceptual speech quality measure (PSQM), and measuring normalizing blocks (MNB), PESQ covers a wider range of noise. [32]

$$PESQ = 4.5 - 0.1 * d_{sym} - 0.0309 * d_{asym} \tag{30}$$

A block diagram is shown below. The clean signal and noisy signal are the input of the system. Firstly, both signals are adapted to the same amplitude level and are filtered by a filter that emulates the common telecommunication channel loss. Secondly, the two signals are compared and aligned in time, and processed by the auditory transform. By then, compute the asymmetric and symmetric disturbance, thus, get the PESQ score.

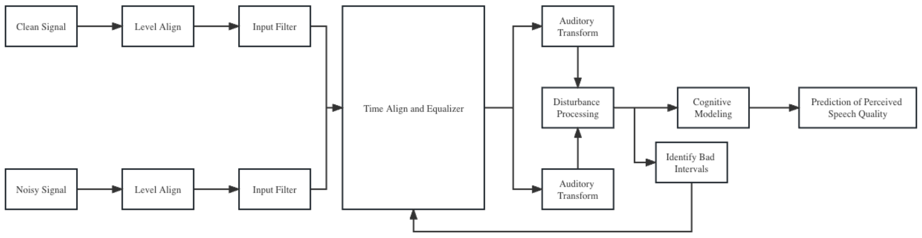

Figure 6: Overall structure of PESQ

The advantage of PESQ is its similarity to the subjective psychoacoustics perception. It successfully covers a number of key elements that represent the overall quality of the degraded audio.

4.4 Experiments and Results

4.4.1 Experiments on Original Dataset

The first set of experiments we performed is using the original testset of the dataset described in section 4.1. We ran the four methods in this testset and calculated the resulting average segSNR and PESQ using the clean speech and filtered noisy speech, and the results are shown in 1.

Method	segSNR	PESQ
Noisy	-3.30	1.96
Wiener Filter	8.87	1.72
LogMMSE	10.81	1.90
NSNET2	15.09	2.51
DeepFilterNet	16.16	3.20

Table 1: The segSNR and PESQ comparison between Wiener Filter, LogMMSE, NSNET2 and DeepFilterNet on original testset

4.4.2 Experiments on Augmented Dataset

The second set of experiments are inspired by [1]. We ran the four methods to test the performance of each using the augmented dataset described in 4.1.

In previous section, we adjusted the noise amplitude and remixed it with the clean signal, to get the noisy signal in different SNR levels. Specifically, the levels are: -5dB, 0dB, 5dB, 7dB, 10dB, 15dB, and 20dB. Our goal is to compare their performance at different SNR levels.

The segSNR results are shown in table 2, and the PESQ results are shown in table 3.
We have also provided spectrograms for a specific signal we picked to given more visually straightforward comparison for the four methods in figure 7.

SNR (dB)	Wiener	LogMMSE	NSNET2	DeepFilterNet
-5	-0.8622	4.0197	8.9363	12.4887
0	3.0023	7.3631	11.3385	14.3849
5	6.5935	10.0221	13.609	15.0403
7	7.9732	10.8661	14.4628	14.896
10	9.9789	11.8507	15.6456	14.8599
15	13.1066	12.8847	17.3322	16.8547
20	15.7826	13.3583	18.6934	20.5572

Table 2: The segSNR comparison between Wiener Filter, LogMMSE, NSNET2 and DeepFilterNet

SNR (dB)	Wiener	LogMMSE	NSNET2	DeepFiltrerNet
-5	1.162	1.3598	1.7762	2.1011
0	1.2768	1.5545	2.0341	2.4668
5	1.456	1.7704	2.3035	2.8508
7	1.5517	1.8604	2.4095	2.9888
10	1.7124	1.9845	2.5576	3.1839
15	2.0313	2.1704	2.7696	3.5008
20	2.3574	2.3129	2.9442	3.7873

Table 3: The PESQ comparison between Wiener Filter, LogMMSE, NSNET2 and DeepFilterNet

4.5 Results Analysis

In this section, we are going analyze our results both qualitatively and quantitatively.

4.5.1 Analysis for 4.4.1

We start by analyzing table 1

- (1) We observed that, all four methods could increase the segSNR, but conventional DSP methods did not do well increasing PESQ. It indicates that although they succeeded in removing noise, they distorted the original signal, thus, affecting the overall quality of it. We sanity checked this abnormal phenomenon by listening to the processed audio, and we found these methods indeed removed noise as well as distorted the original signal.
- (2) Deep Learning methods outperform conventional DSP methods on both metrics, which matches our expectation.
- (3) LogMMSE performs better than the Wiener Filter on both metrics, which matches our expectation.
- (4) DeepFilterNet outperforms NSNET-2 on both metrics, which matches our expectation. SNET-2 has a simpler structure and its mask $G(k,n)$ is a real-valued suppression gain, and it can not recover the signal phase. The DeepFilterNet is much more complex in architecture. It uses deep filtering instead of point-wise multiplication using a real-valued or complex mask. Deep filtering in the STFT domain would allow to incorporate information from previous and future timesteps exploiting local correlations within each frequency band. It first enhanced the

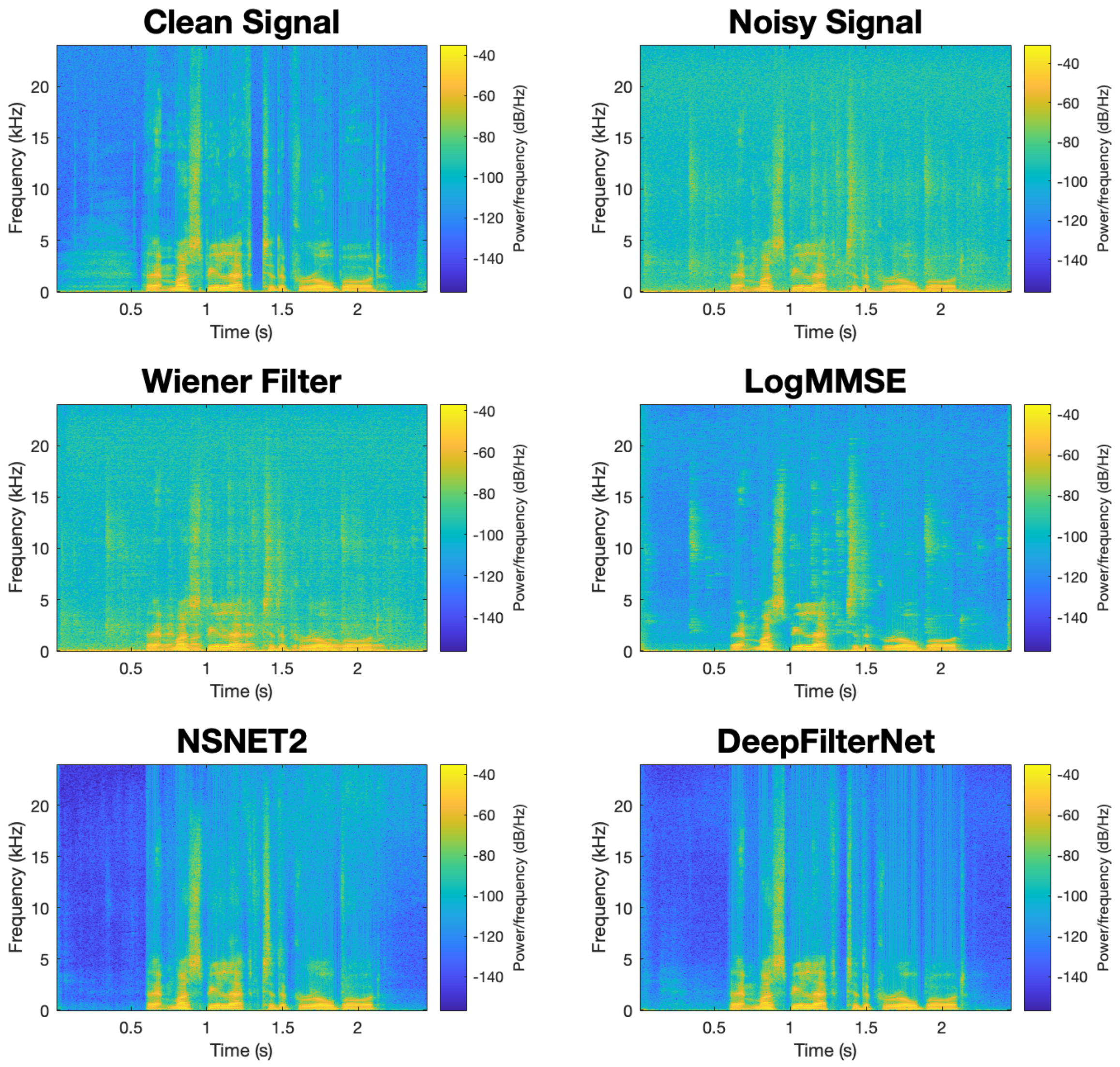

Figure 7: Spectrogram Comparison

spectral envolope using ERB-scaled gains modeling human frequency preception, then utilized deep filtering to enhance the periodic components of the speech.

Lastly, by examining the spectrogram 7, we observed these facts below:

- The LogMMSE filter is good at reducing the stationary noise, such as the additive noise in the background, but failed to recognize the non-stationary noise, especially when the noise level is high.

- The Wiener filter's noise suppression level could be very slight in some cases.

- The NSNET2 can recognize some pieces of non-stationary noise, but failed to recognize most of its parts.

- The DeepFilterNet comprehensively recognized most part of the noise, making the residual noise level extremely small.
- All methods preserves the clean speech spectrogram in the lower frequencies relatively well, but conventional DSP methods performed terrible in the higher frequencies.

4.5.2 Analysis for 4.4.2

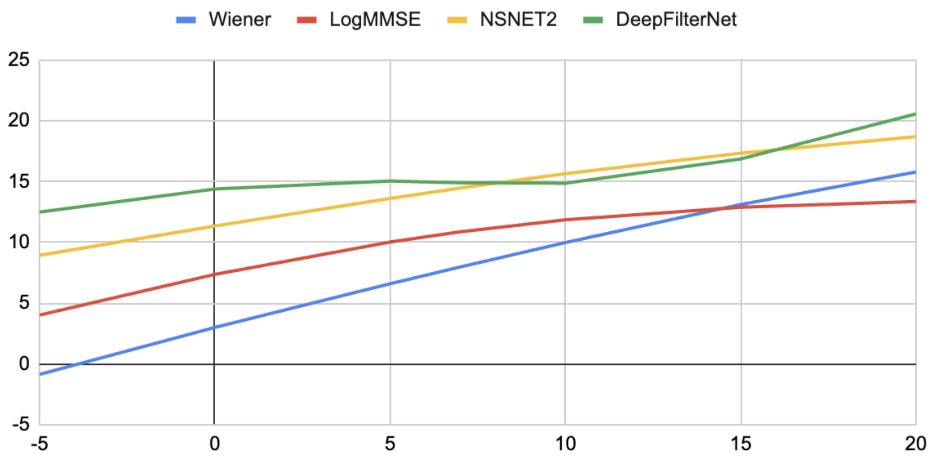

Figure 8: segSNR Comparison

By examining the plot in the figure 8, we observed these facts below:

- The segSNR performance is ranked in this order from high to low: DeepFilterNet, NSNET2, LogMMSE, WienerFilter.

- The performance of DeepFilterNet converges with NSNET2 at 10dB and 15dB, and the performance of LogMMSE converges with Wiener Filter at 15 dB.

- As the SNR increases above 15dB, DeepFilterNet is continuously more supreme than any other algorithms, while the LogMMSE is starting to fall behind others.

We then analyze the plot in the figure 9, we observed these facts below:

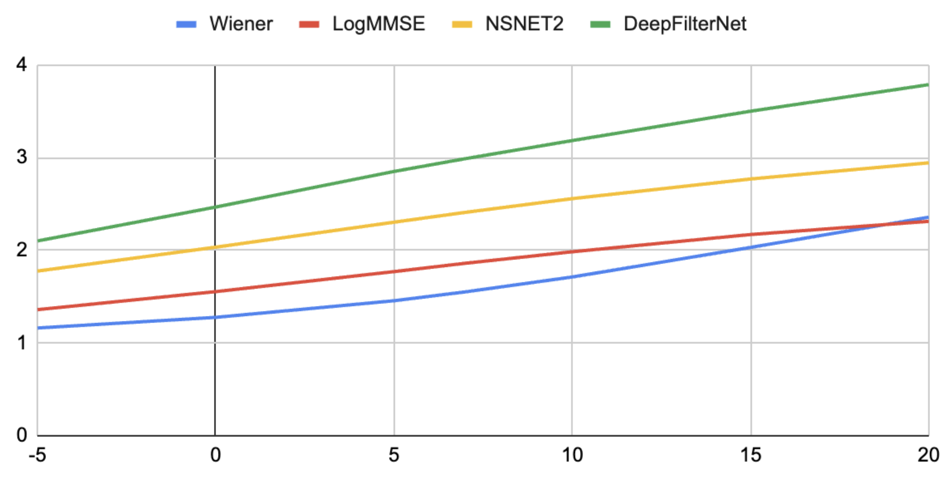

Figure 9: PESQ Comparison

- The PESQ performance is ranked in this order from high to low: DeepFilterNet, NSNET2, LogMMSE, WienerFilter.

- The performance of LogMMSE converges with Wiener Filter at 20 dB.

From table 3 and table 2, we observed that:

- In general, the performance of these four methods on two evaluation metrics can be ranked as: DeepFilterNet > NSNET2 > LogMMSE > Wiener.
- As SNR increases, the segSNR and PESQ results for all four methods generally increase because the input signals have less noise and better overall quality.
- For rule-based methods, when SNR < 10dB, they are effective in in terms of improving the segSNR. When SNR > 10dB, there is not much noise to remove. As a result, these methods tend to distort the signals more rather than remove noise, therefore the overall segSNR of audios degraded.
- For deep learning methods, in terms of segSNR, they consistently perform well until the SNR reaches 20dB. Even at high SNR values, they successfully preserved the original SNR.
- From the PESQ table, we can see the overall quailty of signal gradually increases as the SNR increases, and the performance rank of the four methods is consistent: DeepFilterNet > NSNET2 > LogMMSE > Wiener.

5 Conclusion

5.1 Project Achievements

In this project, we studied the problem of speech enhancement. We chose a well-known dataset and performed augmentation on it to enable more experiments. We explored four speech enhancement techniques, two conventional DSP techniques: Wienner filter, LogMMSE, and two DL techniques: NSNET-2, and DeepFilterNet. We introduced each method's mathematical formulation, network architecture, parameters chosen, and designing concepts behind them. We performed two sets of experiments on the original and augmented datasets and calculated their performance using two evaluation metrics: segSNR and PESQ. We analyzed the results we got both qualitatively and quantitatively.

During this process, we

- got a comprehensive understanding of speech enhancement.
- utilized and reinforced speech processing techiniques and concepts we learned in our ECE 6255 lectures. For example: STFT, Filter Bank, Adaptive Filtering, etc.
- improved our researching and literature reading skills.
- improved our programming skills in MATLAB, Python and PyTorch.
- understood the formal process of researching, from problem formulation to problem solving.
- had good communication and collaboration, and established good friendship.
- learned how write well-formatted report using LaTex.

5.2 Difficulties

In this experiment, we overcome a number of challenges. These challenges can be classified into these types below:

- Heavy workload of research on related works The denoising task has long been researched by a number of researchers, who invented a considerable amount of methods dealing with all kinds of problems. Some classical method, such as the Wiener Filter, even has several dozens of variations, which increased the difficulty of classifying them.
- The Computational Complexity of Programs For the DNN models, it requires a significant amount of time in terms of training and prediction. For some rule-based algorithms, such as the LogMMSE, it is originally using the exponential integral function, which consumes a lot of time. Our method pre-calculates this function with thousands of numbers and loads it when the LogMMSE needs it.
- Mathematical Analysis Some of the math we encountered requires lots of prerequisite knowledge to understand, for example, it took us a very hard time to even understand the equations listed in the DeepFilterNet paper.
- Both of us are new to LaTex, it was painful for us to learn these commands and formats as beginners, but we feel this would be a very useful technique for us in our future study and work.

5.3 Potential future work

Although this project is hard and long for us, we had plenty of fun during this procedure, and we both have developed interests in speech enhancement, and digital signal processing. Although we have learned a lot, there are plenty of other great approaches to explore, and this area is still an active field of research.

In the future, we will keep reading related papers, and dive deeper into this field.

References

[1] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee," An Experimental Study on Speech Enhancement Based on DeepNeural Networks", 2014.

[2] Lan Tian, Peng Chuan, Li Sen, YeWen zheng, Li Meng, Hui Guoqiang, Lü Yilan, Qian Yuxin,and Liu Qia, "An Overview of Monaural Speech Denoising and Dereverberation Research", 2020.

[3] Se Rim Park, Jin Won Lee, "A Fully Convolutional Neural Network for Speech Enhancement", 2016.

[4] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," Journal ofMachine Learning Re- search, vol. 11, no. Dec, pp. 3371–3408, 2010.

[5] Yanxin Hu,Yun Liu,Shubo Lv, Mengtao Xing, Shimin Zhang, Yihui Fu, Jian Wu, Bihong Zhang, Lei Xie, "DCCRN:DeepComplex Convolution Recurrent Network for Phase-Aware Speech Enhancement"

[6] K. Tan and D. Wang, "A convolutional recurrent neural network for real-time speech enhancement." in Interspeech, vol. 2018. 2018, pp. 3229–3233

[7] K. Tan and D. Wang, "Complex spectral mapping with a convolutional recurrent network for monaural speech enhancement," in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 6865– 6869.

[8] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," NASA STI/Recon technical report n, vol. 93, 1993

[9]] J. Garofolo, D. Graff, D. Paul, and D. Pallett, "Csr-i (wsj0) complete ldc93s6a," Web Download. Philadelphia: Linguistic Data Consortium, vol. 83, 1993

[10] Veaux, C.; Yamagishi, J.; King, S. The voice bank corpus: Design, collection and data analysis of a large regional accent speech database. In Proceedings of the 2013 International Conference Oriental COCOSDA Held Jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE), Gurgaon, India, 25–27 November 2013; pp. 1–4.

[11] Thiemann, J.; Ito, N.; Vincent, E. The diverse environments multi-channel acoustic noise database (demand): A database of multichannel environmental noise recordings. Proc. Meet. Acoust. 2013, 19, 035081.

[12] Varga, A.; Steeneken, H.J. Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems. Speech Commun. 1993, 12, 247–251

[13] Stowell, D.; Plumbley, M.D. An open dataset for research on audio field recording archives: freefield1010. arXiv 2013, arXiv:1309.5275

[14] Abd El-Fattah, M. A., Dessouky, M. I., Abbas, A. M., Diab, S. M., El-Rabaie, E.- S. M., Al-Nuaimy, W., Alshebeili, S. A., and Abd El-samie, F. E. Speech enhancement with an adaptive wiener filter. International Journal of Speech Technology 17 (2014), 53–64.

[15] Benesty, J., Sondhi, M. M., Huang, Y., et al. Springer handbook of speech processing, vol. 1. Springer, 2008

[16] Boll, S. Suppression of acoustic noise in speech using spectral subtraction. IEEE Transactions on acoustics, speech, and signal processing 27, 2 (1979), 113–120.

[17] Johnstone, I. M., and Silverman, B. W. Wavelet threshold estimators for data with correlated noise. Journal of the royal statistical society: series B (statistical methodology) 59, 2 (1997), 319–351.

[18] Rix, A. W., Beerends, J. G., Hollier, M. P., and Hekstra, A. P. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In 2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221) (2001), vol. 2, IEEE, pp. 749–752.

[19] Widrow, B., and Hoff, M. E. Adaptive switching circuits. Tech. rep., Stanford Univ Ca Stanford Electronics Labs, 1960

[20] Yelwande, A., Kansal, S., and Dixit, A. Adaptive wiener filter for speech enhancement. In 2017 international conference on information, communication, instrumentation and control (icicic) (2017), IEEE, pp. 1–4

[21] Valentini-Botinhao, Cassia. ”Noisy speech database for training speech enhancement algorithms and tts models.” University of Edinburgh. School of Informatics. Centre for Speech Technology Research (CSTR) (2017)., available at: https://datashare.ed.ac.uk/handle/10283/2791.

[22] Valentini-Botinhao, Cassia, et al. ”Speech Enhancement for a Noise-Robust Text-to-Speech Synthesis System Using Deep Recurrent Neural Networks.” Interspeech. Vol. 8. 2016.

[23] Veaux, Christophe, Junichi Yamagishi, and Simon King. ”The voice bank corpus: Design, collection and data analysis of a large regional accent speech database.” 2013 international conference oriental COCOSDA held jointly with 2013 conference on Asian spoken language research and evaluation (O-COCOSDA/CASLRE). IEEE, 2013.

[24] Thiemann, Joachim, Nobutaka Ito, and Emmanuel Vincent. ”The diverse environments multichannel acoustic noise database (demand): A database of multichannel environmental noise recordings.” Proceedings of Meetings on Acoustics ICA2013. Vol. 19. No. 1. Acoustical Society of America, 2013.

[25] Speech shaped and babble noise: http://homepages.inf.ed.ac.uk/cvbotinh/se/noises/.

[26] Widrow, Bernard, and Marcian E. Hoff. Adaptive switching circuits. Stanford Univ Ca Stanford Electronics Labs, 1960.

[27] NSNET2 GitHub code: https://github.com/NeonGeckoCom/nsnet2-denoiser

[28] DeepFilterNet code: https://github.com/Rikorose/DeepFilterNet

[29] Loizou P C. Speech enhancement: theory and practice[M]. CRC press, 2013

[30] Braun, Sebastian, and Ivan Tashev. ”Data augmentation and loss normalization for deep noise suppression.” Speech and Computer: 22nd International Conference, SPECOM 2020, St. Petersburg, Russia, October 7–9, 2020, Proceedings 22. Springer International Publishing, 2020.

[31] Cho, Kyunghyun, et al. ”On the properties of neural machine translation: Encoder-decoder approaches.” arXiv preprint arXiv:1409.1259 (2014).

[32] Rix, Antony W., et al. ”Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs.” 2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221). Vol. 2. IEEE, 2001.

[33] Ephrat, Ariel, et al. ”Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation.” arXiv preprint arXiv:1804.03619 (2018).

[34] Wilson, Kevin, et al. ”Exploring tradeoffs in models for low-latency speech enhancement.” 2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC). IEEE, 2018.

[35] Schroter, Hendrik, et al. ”DeepFilterNet: A low complexity speech enhancement framework for full-band audio based on deep filtering.” ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022.

[36] Schröter, Hendrik, et al. ”Lightweight online noise reduction on embedded devices using hierarchical recurrent neural networks.” arXiv preprint arXiv:2006.13067 (2020).

[37] Schröter, Hendrik, et al. "Clcnet: Deep learning-based noise reduction for hearing aids using complex linear coding." ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020.

[38] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer International Publishing, 2015.

[39] Boll, Steven. "Suppression of acoustic noise in speech using spectral subtraction." IEEE Transactions on acoustics, speech, and signal processing 27.2 (1979): 113-120.

[40] Johnstone, Iain M., and Bernard W. Silverman. "Wavelet threshold estimators for data with correlated noise." Journal of the royal statistical society: series B (statistical methodology) 59.2 (1997): 319-351.

A

Appendix A. evaluation.m

```
num_files = 127; % how many clips you want to measure
SNR = zeros(num_files, 1);
PESQ = zeros(num_files, 1);

folder_noisy = "../data/noisy_testset_wav";
folder_clean = "../data/clean_testset_wav";
folder_processed = "../data/output/clean_DFN/20dB_clean";
clean_path = ls("../data/clean_testset_wav");
noisy_path = ls("../data/noisy_testset_wav");
processed_path = ls("../data/output/clean_DFN/20dB_clean");
processed_name = split(processed_path);
clean_name = replace(processed_name, "_DeepFilterNet2.wav", ".wav");
noisy_name = replace(processed_name, "_DeepFilterNet2.wav", ".wav");


for i = 1:num_files

    f_clean = folder_clean + "/" + clean_name(i);
    f_noisy = folder_noisy + "/" + noisy_name(i);
    f_processed = folder_processed + "/" + processed_name(i);

    [x_noisy, fs] = audioread(f_noisy);
    [x_clean, ~] = audioread(f_clean);
    [x_processed, ~] = audioread(f_processed);

    tic;
    y = x_processed;
    disp("file clean: " + f_clean);
    disp("file processed: " + f_processed);
    disp("process consumed: " + toc);

    SNR(i) = compute_seg_snr(x_clean, y, "none");
    PESQ(i) = pesq(x_clean, y, fs, "none");

end


avgSNR = mean(SNR);
```

```
avgPESQ = mean(PESQ);
```

B
Appendix B. wienerFilter.m

```
function xhat = wienerFilter(y, n, nfft, noverlap, fs)
    len = length(y);
    Syy = pwelch(y, hann(nfft, "periodic"), noverlap, nfft, fs, "twosided");
    Snn = pwelch(n, hann(nfft, "periodic"), noverlap, nfft, fs, "twosided");
    Hspec = max(0, 1 - abs(Snn./Syy));
    h = ifft(Hspec);
    xhat = fftfilt(h, y);
    xhat = xhat(1:len);
end
```

C
Appendix C. logmmse.m

```
function y = logmmse(input, Srate, noise_profile)

x = input;

% noise_ref = zeros(length(x), 1);
% noise_ref(1:length(noise_profile)) = noise_profile;
if nargin > 2
    len_noise = length(noise_profile);
    x(1:len_noise) = noise_profile;
end

% =============== Initialize variables ===============

len=floor(20*Srate/1000); % Frame size in samples
if rem(len,2)==1, len=len+1; end;
PERC=50; % window overlap in percent of frame size
len1=floor(len*PERC/100);
len2=len-len1; % update rate in samples


win=hanning(len);  % define window
win = win*len2/sum(win);  % normalize window for equal level output


% Noise magnitude calculations - assuming that the first 6 frames is
% noise/silence

nFFT=2*len;
noise_mean=zeros(nFFT,1);
j=1;
for m=1:50
    noise_mean=noise_mean+abs(fft(win.*x(j:j+len-1),nFFT));
    j=j+len;
end
noise_mu=noise_mean/50;
noise_mu2=noise_mu.^2;
```

```
%--- allocate memory and initialize various variables



x_old=zeros(len1,1);
Nframes=floor(length(x)/len2)-floor(len/len2);
xfinal=zeros(Nframes*len2,1);


%===============================  Start Processing ==========================================
%
k=1;
aa=0.98; % shan: although increase alpha usually increase the noise suppresion, 0.98 seems already a
mu=0.98;
eta=0.15; % shan: I manually set eta from 0.15 to 0, avoid the noise_profile updating, given the nois

ksi_min=10^(-25/10); % ksi and gamma affects the noise suppresion, but not significant in this case

% shan: load EXPINT.mat to speedup
load("./EXPINT.mat", "EXP_high");
load("./EXPINT.mat", "EXP_low");

for n=1:Nframes

    insign=win.*x(k:k+len-1);

    spec=fft(insign,nFFT);
    sig=abs(spec); % compute the magnitude
    sig2=sig.^2;

    gammak=min(sig2./noise_mu2,40);  % limit post SNR to avoid overflows
    if n==1
        ksi=aa+(1-aa)*max(gammak-1,0);
    else
        ksi=aa*Xk_prev./noise_mu2 + (1-aa)*max(gammak-1,0);     % a priori SNR
        ksi=max(ksi_min,ksi);  % limit ksi to -25 dB
    end

    log_sigma_k= gammak.* ksi./ (1+ ksi)- log(1+ ksi);
    vad_decision= sum(log_sigma_k)/ len;
    if (vad_decision< eta)
        % noise only frame found
        noise_mu2= mu* noise_mu2+ (1- mu)* sig2;
    end
    % ===end of vad===

    A=ksi./(1+ksi);  % Log-MMSE estimator
    vk=A.*gammak;
    % shan: modified expint
    vk = vk + 0.0001;
    ei_vk = zeros(length(vk), 1);
    idx_low = find(vk<1);
    idx_high = find(vk>=1);
    ei_vk(idx_low) = 0.5 * EXP_low(cast(vk(idx_low)*10000, "uint16"));
    ei_vk(idx_high) = 0.5 * EXP_high(cast(vk(idx_high), "uint16"));
```

```
%     ei_vk=0.5*expint(vk);
    hw=A.*exp(ei_vk);

    sig=sig.*hw;
    Xk_prev=sig.^2;

    xi_w= ifft( hw .* spec,nFFT);
    xi_w= real( xi_w);

    xfinal(k:k+ len2-1)= x_old+ xi_w(1:len1);
    x_old= xi_w(len1+ 1: len);

    k=k+len2;

end

% wavwrite(xfinal,Srate,16,outfile);

% shan: logmmse strongly based on the first few block noises as reference
% if not, better output the original singal
if sum(isnan(xfinal)) > 0
    y = input;
else
    y = input;
    y(1:length(xfinal)) = xfinal;
end

end
```

D
Appendix D. computeSegSnr.m

```
function y = compute_seg_snr(x_clean, x_hat, DNN)
    if DNN == "DeepFilterNet"
        delay = 77393 - 76913;
        x_hat = circshift(x_hat, delay*-1);

        if length(x_clean) > x_hat
            x_clean(length(x_hat)+1:end) = [];
        end
    end

    if DNN == "NSNET2"
        if length(x_clean) > x_hat
            x_clean(length(x_hat)+1:end) = [];
        end
    end

    % standard segment SNR
    N = 0;
    block_length = 4096;
    jump_length = 1024;
    sumSNR = 0;
    while(N*jump_length+block_length < length(x_clean))
```

```
        sumSNR = sumSNR + 10 * log10 (sum(x_clean.^2)/sum((x_hat - x_clean).^2));
        N = N + 1;
    end

    y = sumSNR / (N+1);

end
```

E
Appendix E. pesq.m

```
function scores = pesq( ref_data, deg_data, fs, DNN)

if DNN == "DeepFilterNet"
    delay = 77393 - 76913;
    deg_data = circshift(deg_data, delay*-1);

    if length(ref_data) > deg_data
        ref_data(length(deg_data)+1:end) = [];
    end
end

if DNN == "NSNET2"
    if length(ref_data) > deg_data
        ref_data(length(deg_data)+1:end) = [];
    end
end


if fs == 48000
    ref_data = resample(ref_data, 16000, 48000);
    deg_data = resample(deg_data, 16000, 48000);
elseif fs == 44100
    ref_data = resample(ref_data, 16000, 44100);
    deg_data = resample(deg_data, 16000, 44100);
end

ref_sampling_rate = 16000;
deg_sampling_rate = 16000;

    if ref_sampling_rate ~= deg_sampling_rate, error( '%s.m: Sampling rate mismatch.\nThe sampling ra
    else, sampling_rate = ref_sampling_rate; end;

    if sampling_rate==8E3, mode='narrowband';
    elseif sampling_rate==16E3, mode='wideband';
    else, error( '%s.m: Unsupported sampling rate (%i Hz).\nOnly sampling rates of 8000 Hz (for narro
    end

    clearvars -global Downsample DATAPADDING_MSECS SEARCHBUFFER Fs WHOLE_SIGNAL Align_Nfft Window

    global Downsample DATAPADDING_MSECS SEARCHBUFFER Fs WHOLE_SIGNAL
    global Align_Nfft Window

    setup_global( sampling_rate );
    TWOPI= 6.28318530717959;
    for count = 0: Align_Nfft- 1
        Window(1+ count) = 0.5 * (1.0 - cos((TWOPI * count) / Align_Nfft));
    end

    ref_data= ref_data(:).';
    ref_data= ref_data* 32768;
    ref_Nsamples= length( ref_data)+ 2* SEARCHBUFFER* Downsample;
    ref_data= [zeros( 1, SEARCHBUFFER* Downsample), ref_data, ...
        zeros( 1, DATAPADDING_MSECS* (Fs/ 1000)+ SEARCHBUFFER* Downsample)];

    deg_data= deg_data(:).';
    deg_data= deg_data* 32768;
    deg_Nsamples= length( deg_data)+ 2* SEARCHBUFFER* Downsample;
    deg_data= [zeros( 1, SEARCHBUFFER* Downsample), deg_data, ...
        zeros( 1, DATAPADDING_MSECS* (Fs/ 1000)+ SEARCHBUFFER* Downsample)];

    maxNsamples= max( ref_Nsamples, deg_Nsamples);

    ref_data= fix_power_level( ref_data, ref_Nsamples, maxNsamples);
    deg_data= fix_power_level( deg_data, deg_Nsamples, maxNsamples);


% KKW ---------

    switch lower( mode )

        case { [], '', 'nb', '+nb', 'narrowband', '+narrowband' }

            standard_IRS_filter_dB= [0, -200; 50, -40; 100, -20; 125, -12; 160, -6; 200, 0;...
                250, 4; 300, 6; 350, 8; 400, 10; 500, 11; 600, 12; 700, 12; 800, 12;...
                1000, 12; 1300, 12; 1600, 12; 2000, 12; 2500, 12; 3000, 12; 3250, 12;...
                3500, 4; 4000, -200; 5000, -200; 6300, -200; 8000, -200];

            ref_data= apply_filter( ref_data, ref_Nsamples, standard_IRS_filter_dB);
            deg_data= apply_filter( deg_data, deg_Nsamples, standard_IRS_filter_dB);

        case { 'wb', '+wb', 'wideband', '+wideband' }
            ref_data = apply_filters_WB( ref_data, ref_Nsamples );
            deg_data = apply_filters_WB( deg_data, deg_Nsamples );

        otherwise
            error( sprintf('Mode: "%s" is unsupported.', mode) );

    end

% -------------
```

F
Appendix F

For our deep learning methods, NSNET2 and DeepFilterNet, we did not implement them by ourselves, but used these open source code available on Github. We only made slight modifications and we kindly ask our readers to refer these links:

- NSNET2: https://github.com/NeonGeckoCom/nsnet2-denoiser

- DeepFilterNet: https://github.com/Rikorose/DeepFilterNet