Birla Institute of Technology and Science, Pilani

Department of Electrical and Electronic Engineering

Speech Background Noise Suppression with Deep Learning

Author:
Nipun Agarwal
BITS Pilani, Pilani Campus

Supervisor:
Roberto G. Valenti
MathWorks

An Assignment submitted for the SJTU EE397:

MathWorks Excellence in Innovation Projects

https://github.com/NipunAgarwal16/Speech-Background-Noise-Suppression-with-Deep-Learning

May 9, 2021

Contents

1 Introduction

1.1 Problem Background

WTO data shows that 1.5 billion people worldwide suffer from some degree of hearing loss. Among those who can benefit from the use of hearing aids, only 17% use hearing aids. Background noise is very detrimental to these devices because it reduces the clarity and quality of speech.

In addition, since the outbreak of the epidemic, there have been more and more online meetings and remote offices, and certain background noises have been generated due to factors such as the network, equipment, and environment, resulting in a very poor experience for users. Hence,noise suppression is necessary for our work and daily life.

1.2 Specific Requirement

- Address all kinds of noise, especially in the case of low signal-noise ratio
- Keep the integrity of the voice information while removing the noise.

1.3 Technological Method

Noise suppression methods based on artificial intelligence (AI) or deep learning have shown promising results. We are able to design and train artificial intelligence noise suppression models whose performance may exceed more traditional signal processing techniques.

1.4 Potential Application

- Hearing aid
- Smart device
- Online meeting
- Remote control

2 Related Works

In order to reduce noise, existing methods can be divided by whether clean speech signals are used or not. Traditional method without clean data and deep learning network using Wavelet threshold denoising and Wiener filtering [1] can effectively reduce noise, but it has a high time complexity. Another method called Noisy2Noisy signal mapping [2] uses two noisy realizations of the same speech signal as the input and the target of a fully convolutional neural network. This solution is suitable for denoising noisy speech signals in real-world audio environments by not requiring to have clean speech signals.

Clean-need methods generally use a deep learning network for denoising. In order to get the input data, some methods use STFT as a pretreatment. Nils L. Westhausen proposes DTLN [3], using STFT to pretreat and LSTM to train, which works robustly in noisy reverberant environments. Umut Isik proposes PoCoNet [4], a large U-Net with DenseNet and self-attention blocks with frequency-positional embeddings, which also use STFT to get the input data. Another kind of methods take the phase of the speech signals into account. Dario Rethage use Wavenet [5], makes use of non-causal, dilated convolutions and predicts target fields instead of a single target sample, taking advantage of the phase of signals. Phase-aware single-stage speech denosing [6] proposed by Hyeong-Seok Choi also used the phase. This method uses a new masking method called phase-aware β-sigmoid mask (PHM), which reuses the estimated magnitude values to estimate the clean phase. Except the methods above, some methods use a special pretreatment. Jean-Mare Valin proposes a hybrid DSP/deep learning approach [7]. This approach extracts the MFCC of the speech signals and the relation of the clean data and noisy data as input and output of the RNN network, greatly reduces the time complexity. In our report, we mainly refer to the last method, and make a little bit of modification on that basis.

3 Preparation

Our project is based on the following environment:

- **Software environment:** MATLAB R2021a
- **Speech data set:** Microsoft DNS challenge repository: https://data.solak.de/data/Training/stt_tts/en_UK.tgz

  A total of 10,000 pieces of voice data are actually used, the language is English, and the length of each piece of voice data ranges from 5s to 20s, and the sampling rate is uniformly set to 8KHz.
- **Training environment:** Personal laptops and SJTU HPC Studio.
- **Hardware development:** Raspberry Pi 4B

4 System Design

4.1 System Overview

Our system is based on the structure of RNNoise, and some changes have been made. The overall structure of the system is shown in Figure 1.

4.2 Training Network

The training process of the network is shown in Figure 2.

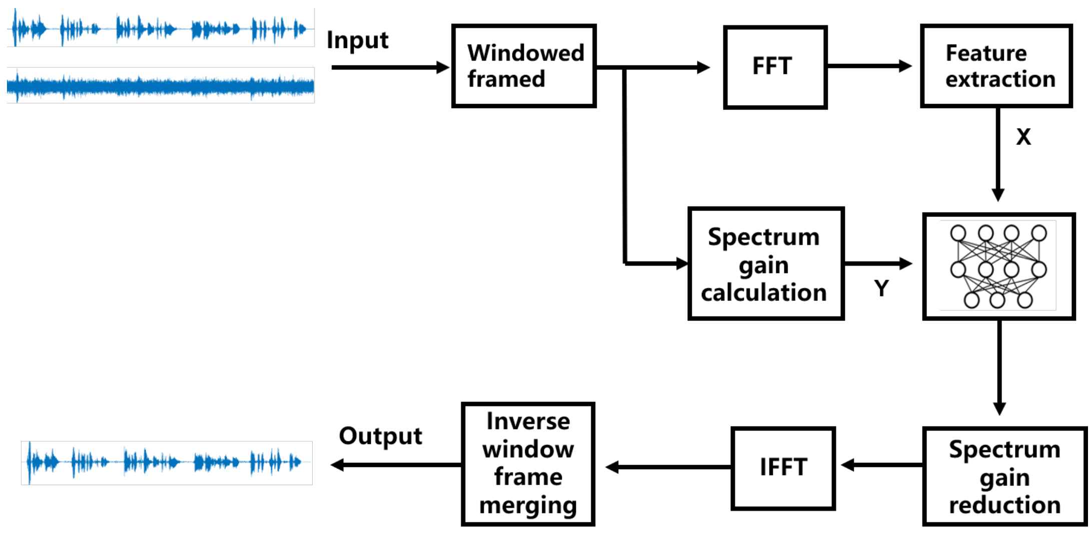


Figure 1: system overview

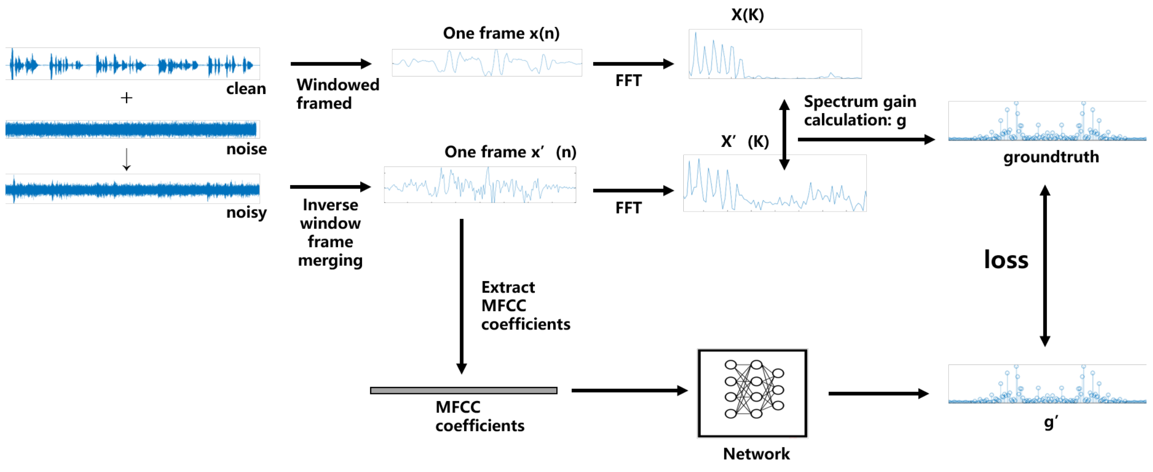


Figure 2: training process

First, perform windowing and framing processing on the clean speech and the same noisy speech with Gaussian noise, perform FFT on each frame of data to obtain the frequency spectrum of each frame, and calculate the noisy speech through the frequency spectrum of the noisy speech and the clean speech. The spectral gain of speech relative to clean speech is used as the groundtruth of the network.

At the same time, MFCC coefficients are calculated for each frame of noisy speech data as the input of the network.

The deep neural network uses the MFCC coefficients of noisy speech to output the spectral gain of the corresponding frame for clean speech.

4.3 Test Network

After training, the process of testing/actual use of the network is shown in Figure 3.

For a new piece of noisy speech, similar to the training, the windowed and framing opera-

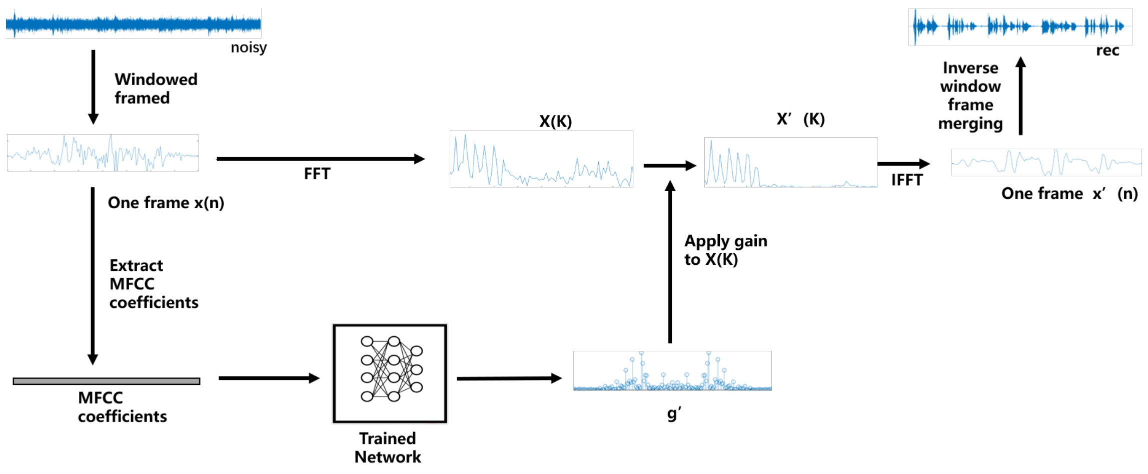


Figure 3: test process

tions are first performed, and then for each processed frame, the MFCC coefficient of the frame is calculated and input to the trained neural network, and the neural network outputs the spectral gain value of the frame.

Meanwhile, FFT is performed on this frame to obtain the frequency spectrum of the frame, and the spectrum gain of the network output is applied to the frequency spectrum of the noisy speech frame to obtain the reconstructed clean speech frame frequency spectrum, and then IFFT is performed to restore the time domain data of the frame.

After the operation of each frame is completed, we performd inverse window and frame overlap restoration to obtain the reconstructed denoised speech.

4.4 Windowed and Framed

The voice signal has time-varying characteristics, but its characteristics remain basically unchanged in a short time range, that is, relatively stable, so it can be regarded as a quasi-steady state process, that is, the voice signal has short-term stability.

In order to make the feature parameters change more smoothly, some frames are inserted between two non-overlapping frames to extract the feature parameters, which forms the overlapping part between adjacent frames.

In specific implementation, add a first-order sine window to each frame of speech:

$$a(m) = sin(\frac{\pi(m+1)}{N+1}) \tag{1}$$

The specific parameters of windowed and framed are as follows:

- Audio sampling rate: 8KHz
- Frame length: 20ms(160 points)
- Overlap between frames: 10ms(80 points)

4.5 Feature Extraction

In order to reduce the complexity of the network input, while keeping the original data characteristics as much as possible, it is necessary to extract the characteristics of the speech completed by the frame, which is specifically implemented to calculate the MFCC coefficients of each frame of speech.

Mel-scale Frequency Cepstral Coefficients (MFCC) is a cepstral parameter extracted in the frequency domain of the Mel scale. The Mel scale describes the non-linear characteristics of the frequency of the human ear. The relationship between it and the frequency can be used under The formula approximates:

$$Mel(f) = 2595 \times lg(1 + \frac{f}{700}) \tag{2}$$

The calculation process of the Mel frequency cepstral coefficient is shown in Figure 5. Since the previous process has been implemented during windowing and framing, this chapter mainly introduces the Mel filter bank, Logarithmic energy calculation and the DCT.

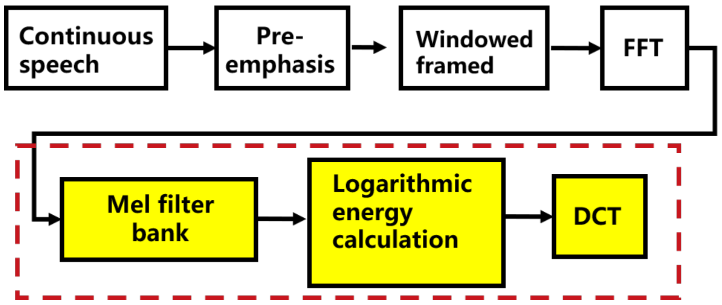


Figure 4: MFCC coefficient calculation process

The Mel filter bank is defined as follows:

$$H_m(k) = \begin{cases} 0 & ,k < f(m-1) \\ \frac{2(k-f(m-1))}{(f(m+1)-f(m-1))(f(m)-f(m-1))} & ,f(m-1) < k < f(m) \\ \frac{2(f(m+1)-k)}{(f(m+1)-f(m-1))(f(m)-f(m-1))} & ,f(m) < k < f(m+1) \\ 0 & ,k > f(m+1) \end{cases} \tag{3}$$

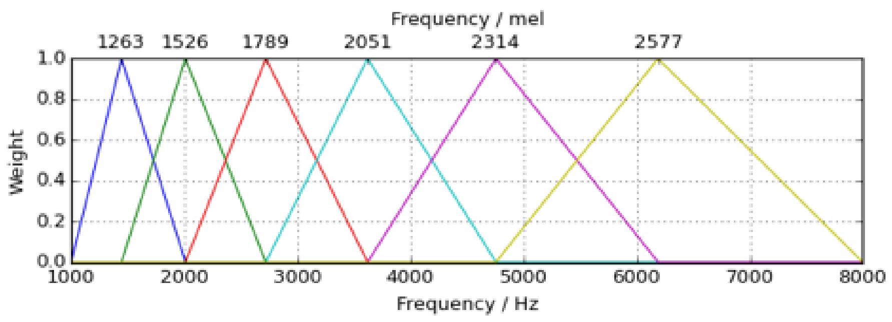


Figure 5: Mel filter bank

After the Mel filter bank, the logarithmic energy calculation and DCT change are performed, as shown in the following:

$$s(m) = ln(\sum_{k=0}^{N-1} |X_a(k)|^2 H_m(k)), \quad 0 \leq m \leq M \tag{4}$$

$$c(n) = \sum_{m=0}^{N-1} s(m) cos(\frac{\pi n(m-0.5)}{M}), \quad n = 1,2,...,L \tag{5}$$

In order to reflect the dynamic characteristics of speech, MFCC differential coefficients can also be added for training.

$$d_t = \begin{cases} C_{t+1} - C_t, t < K \\ \frac{\sum_{k=1}^{K} k(C_{t+k} - C_{t-k})}{\sqrt{2\sum_{k=1}^{K} k^2}} \\ C_t - C - t - 1, t > L - K \end{cases} \tag{6}$$

In the actual model calculation, for each frame:

- MFCC coefficient order $L = 16$
- MFCC differential coefficient order=8

So the input feature of each frame is $16 + 8 = 24$ points.

4.6 Spectrum Gain Calculation

For spectrum gain calculation, there are two feasible methods in this system, namely calculating the gain by frequency and calculating the gain by band energy.

If calculate the gain for the frequency point, we set clean speech frame spectrum $X(k)$, and noisy speech frame spectrum $X^{'}(k)$, so the gain g_k is:

$$g_k = \frac{X(k)}{X'(k)} \tag{7}$$

The gain dimension at this time, that is, the dimension of the network output is the same as the frame length, the gain calculation is more accurate but the calculation amount is relatively large.

If the gain is calculated according to the energy of the frequency band, then the frequency band definition in the Opus diagram can be used for reference, as shown in Figure 6.

The gain at this time is:

$$g_b = \sqrt{\frac{E_s(b)}{E_x(b)}} \tag{8}$$

$E_x(b)$ is is the energy of the b-th frequency band of the clean speech frame spectrum and $E_s(b)$ is the noisy one, when:

$$E(b) = \sum_k w_b(k)|X(k)|^2 \tag{9}$$

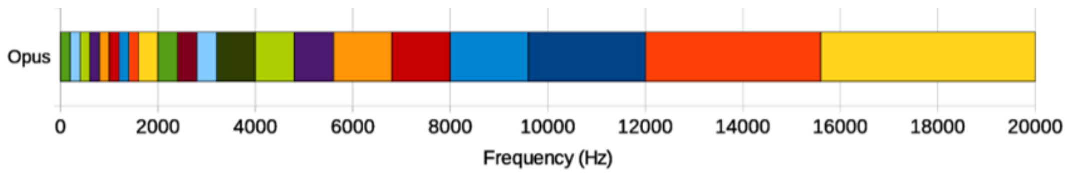


Figure 6: Opus frequency

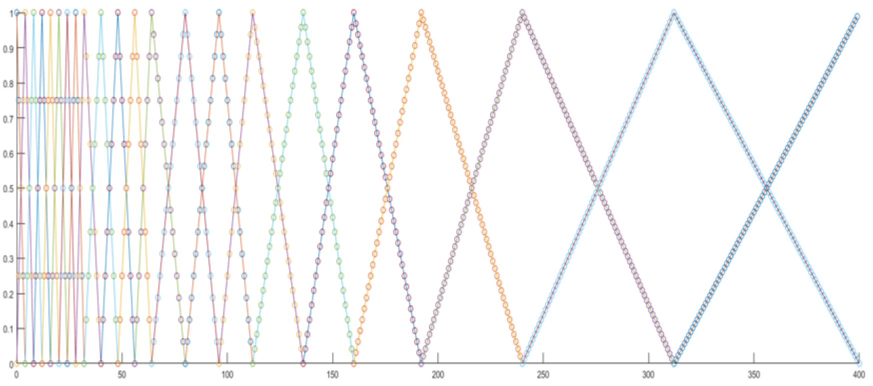


Figure 7: define of w_b

w_b is a set of triangular filters, Satisfy $\sum_b w_b(k) = 1$, defined as shown in Figure 7.

In the actual model calculation, the noise reduction effect of calculating the gain according to the frequency band is slightly worse than the frequency point in all aspects, but the calculation amount is significantly reduced, so the frequency band method is mainly selected, and the frequency point method is used as a comparison. Combined with the sampling rate of the data set, the number of frequency bands is defined as 17.

4.7 Network Design

Due to the superior performance of the time series processing related network on the voice signal, our network draws on the structure of RNNoise that uses the gated recurrent unit (GRU) as the main body. The network structure is shown in Figure 8.

Among them, the input of the network is MFCC and its differential coefficient, and the output is the band gain.

4.8 Spectrum Gain Reduction

After obtaining the ideal frequency band gain g_b of the network output, we can interpolate it to the dimension of the frame and get the denoise spectrum.

$$r(k) = \sum_b w_b(k) g_b \tag{10}$$

$$X(k) = X^{'}(k) \times r(k) \tag{11}$$

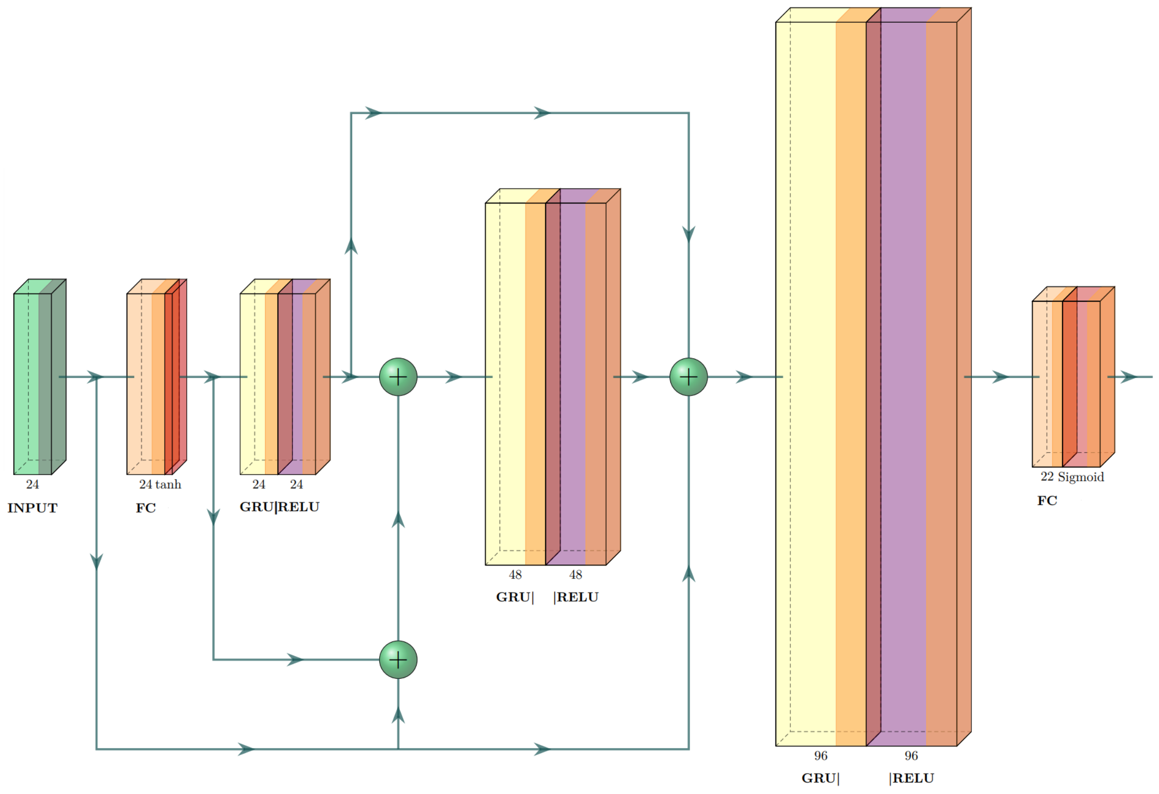


Figure 8: Network Structure

Transform the reconstructed X(k) to the time domain by IFFT to obtain the reconstructed frame data in the time domain.

4.9 Reverse Window Frame Restoration

Since a sine window is added to the voice data during framing, and the frame length is twice the frame shift, at this time, when restoring, using the characteristics of the sine window, you can directly overlap and add the frames to obtain the reconstructed voice data.

The window summary is:

- A window is added before the positive transformation to suppress the side lobes on the spectrum;
- After the inverse transformation and before the overlap addition, a window is added to avoid abrupt changes in the signal after synthesis;
- The sine window multiplied by the sine window is exactly equal to the Hanning window, and the overlap of the half of the Hanning window is exactly equal to 1, so if we do not do any processing after the forward transformation and directly inverse transformation synthesis, we can get the original signal.

5 Evaluation

5.1 Denoise Effect

In this chapter, we will measure the noise reduction effect of our system from many aspects. The first is the most intuitive measurement-the performance of time-domain data and the spectrogram graph.

In training, our noise is set to SNR=0dB, so we first test the noise reduction performance under the condition of 0dB noise. The results of time domain data and spectrogram are shown in Figure 9.

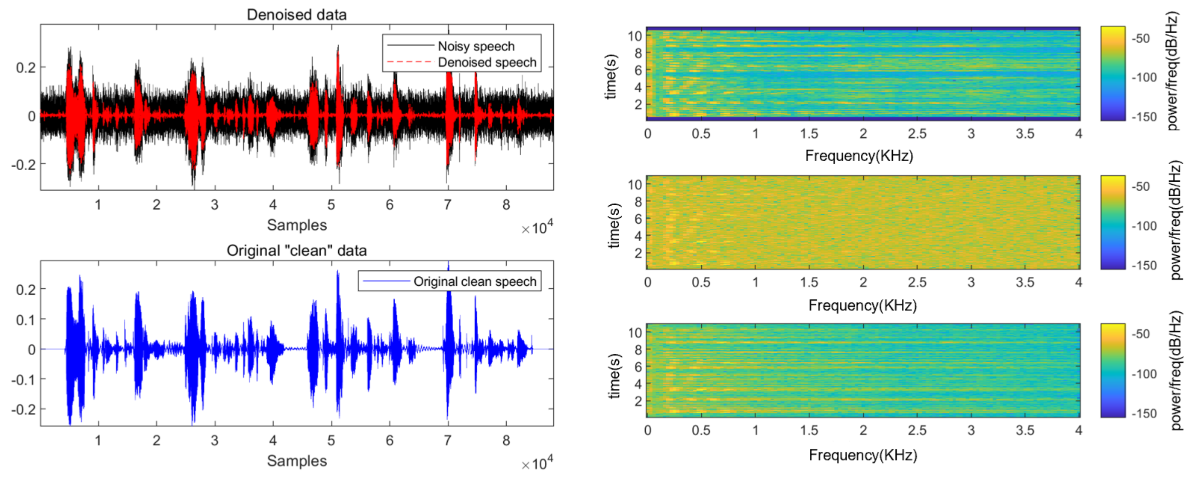


Figure 9: denoise effect in SNR=0dB

It can be seen that the noise reduction effect at this time is better. From the spectrogram, except for the obvious distortion in the high-frequency stage, the remaining parts basically restore the characteristics of the original speech.

Under the condition of weak noise (SNR=5dB), the noise reduction effect is further enhanced, shown in Figure 10.

Under the condition of severe noise (SNR=-5dB), the system can also reduce the influence of noise to a certain extent, but the distortion of the original voice is more serious at this time, shown in Figure 11.

5.2 Comparison of Objective Indicators

5.2.1 Introduction of Indicators

1. **SNR:** The ratio of the power of signals and noises. The definition is as following:

$$SNR = 10\log_{10}\frac{\sum_{n=0}^{L}x^2(n)}{\sum_{n=0}^{L}(x^2(n)-\hat{x}^2(n))} \tag{12}$$

   among which L is the length of the signal, $x^2(n)$is the clean speech signal, and $\hat{x}^2(n)$is the noisy or processed signal.

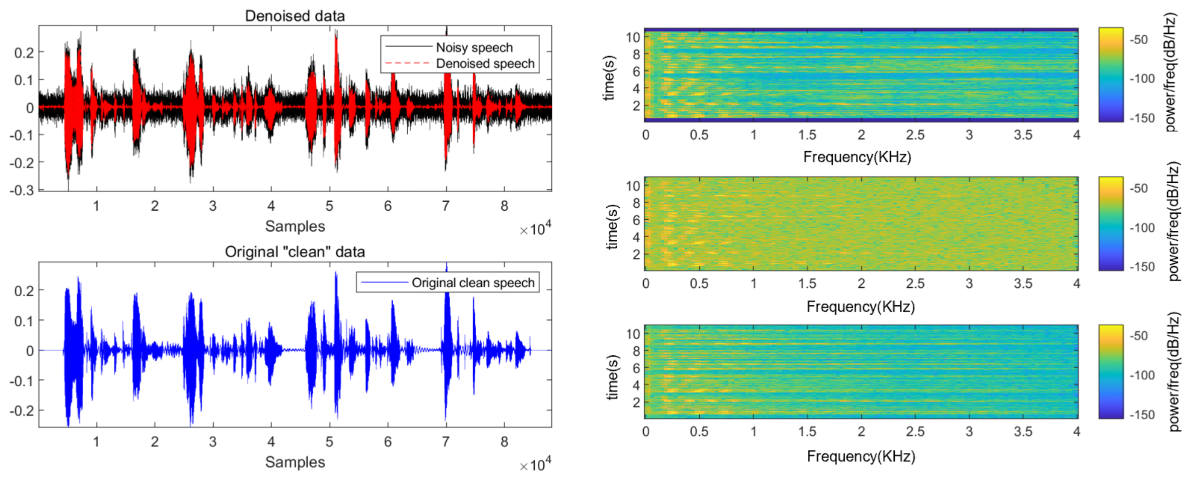


Figure 10: denoise effect in SNR=5dB

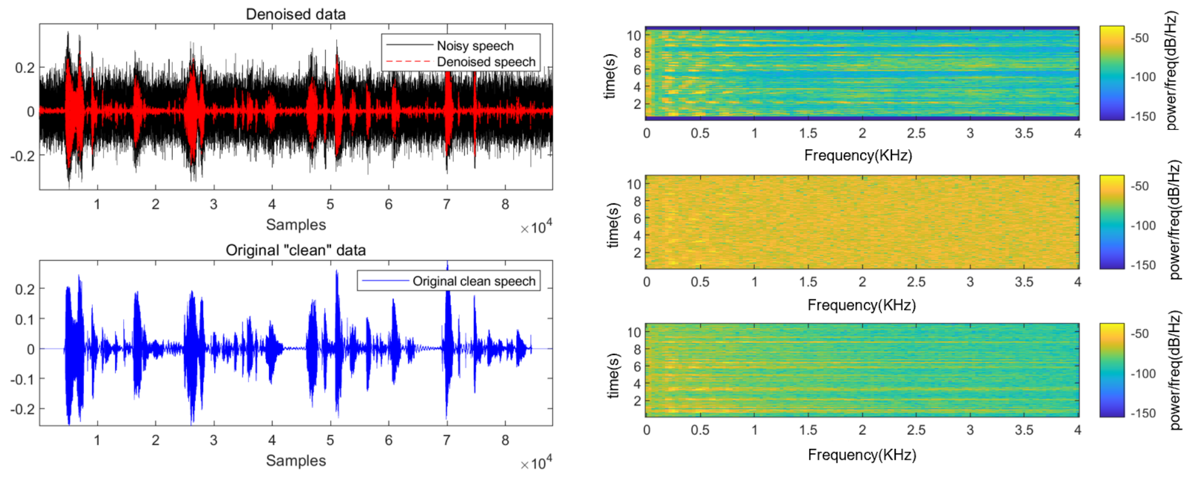


Figure 11: denoise effect in SNR=-5dB

2. **SSNR:** Segmental SNR, the average SNR of per frame. Both SNR and SSNR range from $-\text{inf}$ to $+\text{inf}$, and indicate a closer relationship to the original signal when they are bigger. They reflect the overall effect and has a high correlation with the listener's subjective auditory perception.

3. **LSD:** A measure of the distance between two spectra, ranging from 0 to $+\text{inf}$, and indicate a closer relationship to the original signal when it is smaller.

4. **PESQ:** An objective indicator which is close to the subjective evaluation score (MOS), ranging from -0.5 to 4.5, and indicate a closer relationship to the original signal when they are bigger.

5. **STOI:** Since a word in a speech signal can only be understood or not understood, intelligibility can be regarded as binary from this perspective. Therefore, STOI ranges from 0 to 1, representing the percentage of words correctly understood, and 1 indicating that speech can be fully understood.

5.2.2 Evaluation of Different Methods

We compared our system with the origin method, fully-connected method, convolution method and frequency bin method by indicators mentioned upfront. Each value needs two speech signals to calculate. The noisy signal and clean signal are used to get the indicators before the treatment, and the processed signal and clean signal are used to get the indicators after the treatment. The results are shown in Figure 12.

From the results, it can be inferred that the best performance is produced by frequency bin method, and the second one is our system. The origin method has a pretty PESQ, but its SNR is even worse than SNR before the process. This is because PESQ is a indicator closer to the listening feel, but SNR is just the ratio of power.

5.3 Real Time

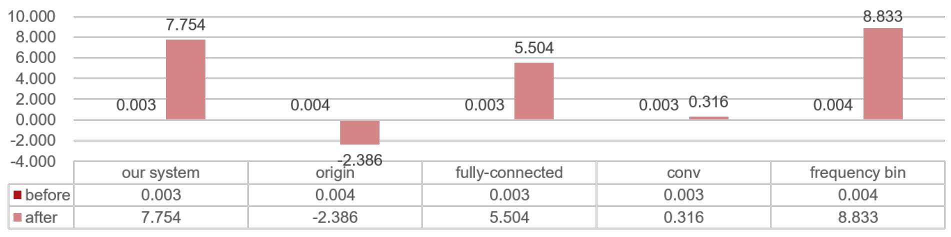
In the project requirements, it is required that our system should take less than the frame stride time to process an input frame of speech otherwise, our system will drop audio samples, degrading the speech quality or rendering it non-intelligible.

So in this unit, we consider the difference in noise reduction processing time between our system and the system participating in the comparison. It should be noted that the real-time test is carried out on a personal computer under the same conditions, and the testing CPU is AMD Ryzen R7-5800H.

We first compare the difference between frame processing time and frame stride, as shown in Figure 13.

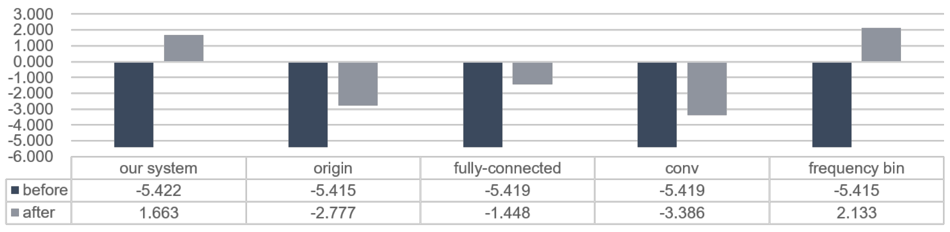
It can be seen that the frame processing time of our system is only about one-fifth of the frame step, which meets the real-time requirements.

At the same time, we can compare the processing time difference of different methods for the same speech, shown in Figure 14.



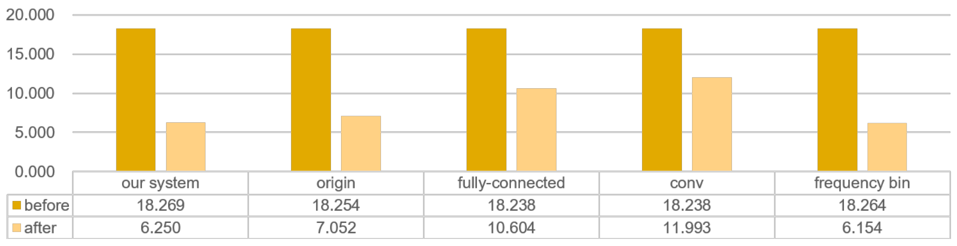
	our system	origin	fully-connected	conv	frequency bin
■ before	0.003	0.004	0.003	0.003	0.004
■ after	7.754	-2.386	5.504	0.316	8.833

(a) SNR



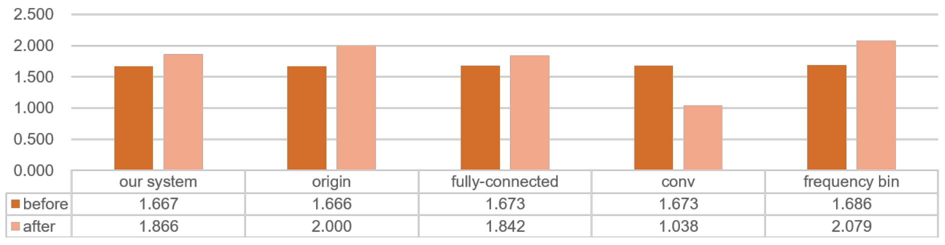
	our system	origin	fully-connected	conv	frequency bin
■ before	-5.422	-5.415	-5.419	-5.419	-5.415
■ after	1.663	-2.777	-1.448	-3.386	2.133

(b) SSNR



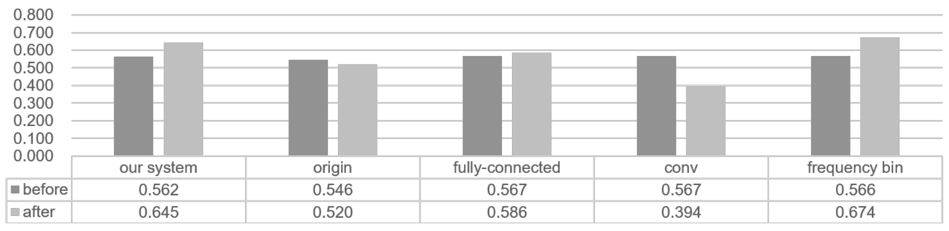
	our system	origin	fully-connected	conv	frequency bin
■ before	18.269	18.254	18.238	18.238	18.264
■ after	6.250	7.052	10.604	11.993	6.154

(c) LSD



	our system	origin	fully-connected	conv	frequency bin
■ before	1.667	1.666	1.673	1.673	1.686
■ after	1.866	2.000	1.842	1.038	2.079

(d) PESQ



	our system	origin	fully-connected	conv	frequency bin
■ before	0.562	0.546	0.567	0.567	0.566
■ after	0.645	0.520	0.586	0.394	0.674

(e) STOI

Figure 12: system real-time

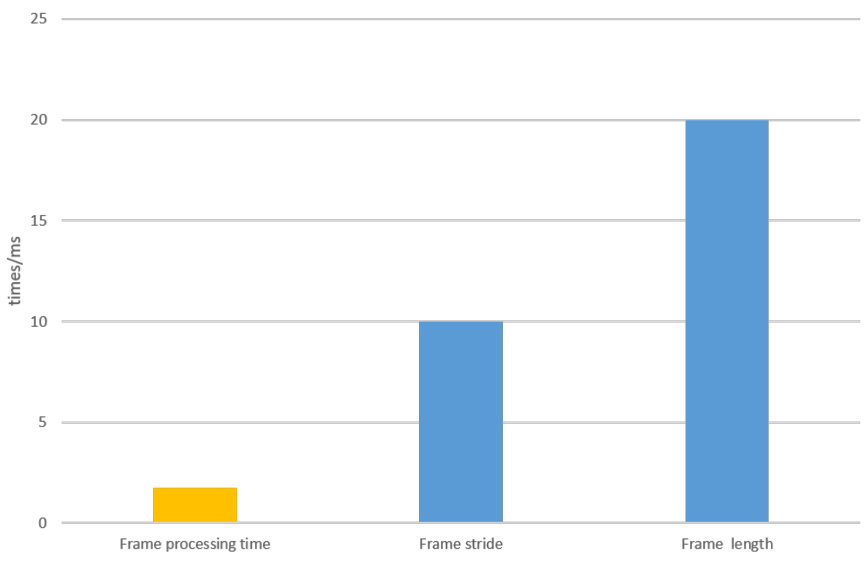


Figure 13: system real-time

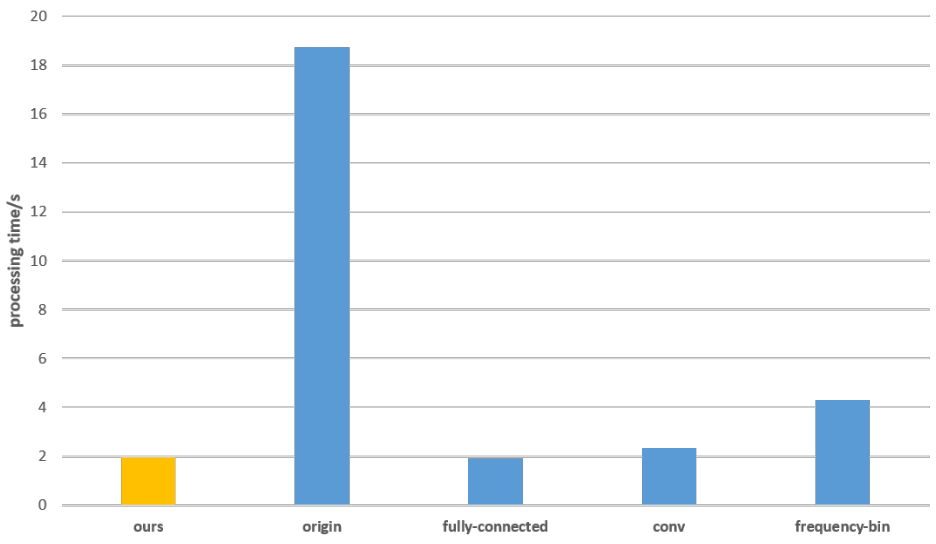


Figure 14: processing time comparison

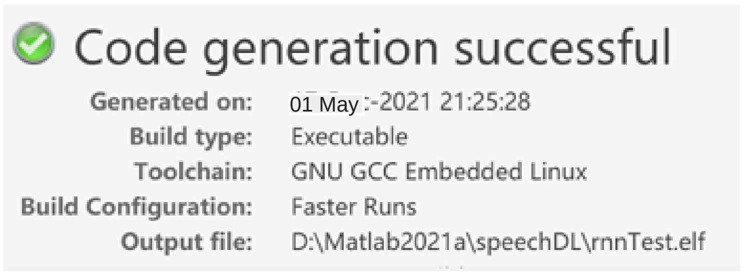
6 Feature Works

In addition to perform real time noise cancellation on MATLAB in real time, we plan to deploy this program on Raspberry Pi 4, so as to achieve this functionality with less computational resource while maintaining its real-time processing.

The idea is to train the network on our computer, which consumes a lot of computational resources, and import the trained network coefficients into the Pi board, since network evaluation costs far less energy.

By using the MATLAB Coder Toolbox, we can generate C++ code from the original MATLAB code, and compile it on the Pi board to produce executable files.

We used some basic functions in the MATLAB code, like `audioread`, `melbankm`, etc. But those fucntions are often not supported by MATLAB Coder, hence cannot be translated into C++ code. We implemented some workaround for that. For `melbankm` which can calculate the Melbank coefficients, we hardcoded that part into our code. For `audioread` and `loadMatrix` to load original audio file and the trained network parameters into the system, we also decided to hardcode that part into our code, just for demonstration purpose. To fully implement this function, we need to edit the generated C++ code accordingly, which we currently have no time to carry out.


Code generation successful

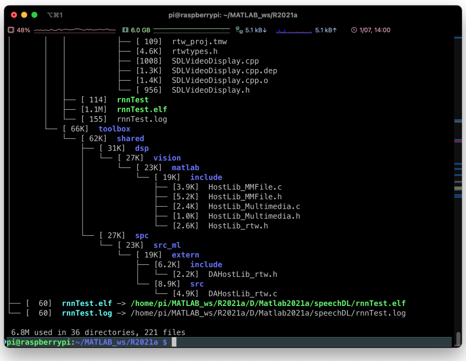
Generated on: 01 May :-2021 21:25:28
Build type: Executable
Toolchain: GNU GCC Embedded Linux
Build Configuration: Faster Runs
Output file: D:\Matlab2021a\speechDL\rnnTest.elf

Figure 15: Generate C++ code for Raspberry Pi

As can be seen, we have successfully generated code for Pi board. An average generation costs us about 8 minutes. With some effort, it can successfully run on the board. But we haven't figure out a way to let the program output its results yet. Since we are not capable of editing the C++ source code for now, we tried to play the corresponding noise-suppressed sound through the 3.5mm headphone jack. Unfortunately, it threw a run-time error. We will keep trying to implement this function in the near future.

7 Conclusion

In conclusion, with the help of the MATLAB platform and the idea of RNNoise, we have implemented an audio noise reduction system based on the combination of deep neural networks and traditional signal processing, and from an intuitive point of view, a variety of objective evaluation indicators and real-time evaluation of the effect of the system, compared with a variety of different types of noise reduction methods. At the same time, our system has a better performance in real-time processing



```
│   │ │     ├── [ 109]  rtw_proj.tmw
│   │ │     ├── [4.6K]  rtwtypes.h
│   │ │     ├── [1008]  SDLVideoDisplay.cpp
│   │ │     ├── [1.3K]  SDLVideoDisplay.cpp.dep
│   │ │     ├── [1.4K]  SDLVideoDisplay.cpp.o
│   │ │     └── [ 956]  SDLVideoDisplay.h
│   │ ├── [ 114]  rnnTest
│   │ ├── [1.1M]  rnnTest.elf
│   │ └── [ 155]  rnnTest.log
│   └── [ 66K]  toolbox
│     └── [ 62K]  shared
│       ├── [ 31K]  dsp
│       │ └── [ 27K]  vision
│       │   └── [ 23K]  matlab
│       │     └── [ 19K]  include
│       │       ├── [3.9K]  HostLib_MMFile.c
│       │       ├── [5.2K]  HostLib_MMFile.h
│       │       ├── [2.4K]  HostLib_Multimedia.c
│       │       ├── [1.0K]  HostLib_Multimedia.h
│       │       └── [2.6K]  HostLib_rtw.h
│       └── [ 27K]  spc
│         └── [ 23K]  src_ml
│           └── [ 19K]  extern
│             ├── [6.2K]  include
│             │ └── [2.2K]  DAHostLib_rtw.h
│             └── [8.9K]  src
│               └── [4.9K]  DAHostLib_rtw.c
├── [  60]  rnnTest.elf -> /home/pi/MATLAB_ws/R2021a/D/Matlab2021a/speechDL/rnnTest.elf
└── [  60]  rnnTest.log -> /home/pi/MATLAB_ws/R2021a/D/Matlab2021a/speechDL/rnnTest.log

6.8M used in 36 directories, 221 files
pi@raspberrypi:~/MATLAB_ws/R2021a $
```

Figure 16: File structure for parts of the generated code

8 Speech Assistant

In addition to the project, we tried to combine it with the voice assistant. We mainly tried the voice assistant based on Baidu AI platform. Below are some implementation details.

8.1 Audio Parameter Concept

- Sampling rate: Baidu speech recognition generally only supports a sampling rate of 16000. That is, 16,000 sampling times per second.
- Bit depth: Lossless audio formats pcm and wav can be set, Baidu speech recognition uses 16bits little endian, that is, 2 bytes to record 1/16000 s of audio data.
- Channel: Baidu speech recognition only supports mono. Take a pcm file encoded with 16bits at a sampling rate of 16000 as an example. Each 16bits (=2bytes) records 1/16000s of audio data. That is, the audio data of 1s is 2bytes * 16000 = 32000B

8.2 Audio File Transcoding

- pcm (not compressed), also known as raw format. Audio input is the most primitive format, no need to decode.
- wav (no compression, pcm encoding): Add a byte describing the sampling rate, encoding and other information at the beginning of the pcm file.
- amr (lossy compression format), which performs lossy compression on audio data, similar to mp3 files.
- m4a (lossy compression format, AAC encoding), lossy compression of audio data, usually only in the format used by WeChat applets. Self-conversion is more complicated.

As the bottom layer recognition uses pcm, it is recommended to upload the pcm file directly. If you upload other formats, it will be transcoded into pcm on the server side, and the time consumption of calling the interface will increase.

8.3 Speech Recognition

Baidu short speech recognition can recognize audio under 60 seconds as text. It is suitable for scenarios such as voice dialogue, voice control, and voice input.

- Interface type: a general HTTP interface provided by way of REST API. Suitable for any operating system, any programming language.
- Interface limitation: The complete recording file needs to be uploaded, and the recording file duration does not exceed 60 seconds. The browser cannot directly call the API interface because it cannot request the domain name of the Baidu voice server across domains.
- Support audio formats: pcm, wav, amr, m4a.
- Audio coding requirements: sampling rate 16000, 8000, 16 bit depth, mono (audio format viewing and conversion).

Call flow:

- Create an account and application: In the `ai.baidu.com` console, create an application and check to enable the "Voice Technology"-"Short Speech Recognition, Short Speech Recognition Speed Edition" capability. Obtain AppID, API Key, Secret Key, and exchange for token through request authentication interface. For details, please refer to "Access Guide".
- Create recognition request: POST method, audio can be submitted in two ways, JSON and RAW. JSON audio data will increase by 1/3 due to base64 encoding. Others fill in specific request parameters, please refer to "Request Description" for details.
- Short speech recognition request address: `http://vop.baidu.com/server_api`
- Return the recognition result: The recognition result will be returned immediately and encapsulated in JSON format. If the recognition is successful, the recognition result will be placed in the "result" field of JSON, and will be encoded in utf-8. See "Return Instructions" for details.

8.4 Speech Synthesis

This paragraph is a guide for the Baidu Voice Open Platform Android SDK, which describes the instructions for using online synthesis, offline synthesis and other related interfaces. The synthesis strategy is to download and play. Different from Rest API downloading the entire recording file at one time. The pure offline speech synthesis SDK can directly perform speech synthesis on the device terminal. It needs to be connected to the Internet for the first use, and offline synthesis can be used in the rest of the time, providing you with a stable, consistent, smooth and natural synthesis experience. The offline speech synthesis SDK needs to apply for SN. After filling the SN into the SDK, the authorization file will be automatically downloaded for the first time online. Offline synthesis of TtsMode. MIX and TtsMode.OFFLINE requires that the authorization file has not expired.

References

[1] Y. Ma and A. Nishihara, "A modified wiener filtering method combined with wavelet thresholding multitaper spectrum for speech enhancement," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2014, no. 1, pp. 1–11, 2014.

[2] N. Alamdari, A. Azarang, and N. Kehtarnavaz, "Improving deep speech denoising by noisy2noisy signal mapping," *Applied Acoustics*, vol. 172, p. 107631, 2021.

[3] N. L. Westhausen and B. T. Meyer, "Dual-signal transformation lstm network for real-time noise suppression," *arXiv preprint arXiv:2005.07551*, 2020.

[4] U. Isik, R. Giri, N. Phansalkar, J.-M. Valin, K. Helwani, and A. Krishnaswamy, "Poconet: Better speech enhancement with frequency-positional embeddings, semi-supervised conversational data, and biased loss," *arXiv preprint arXiv:2008.04470*, 2020.

[5] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5069–5073, IEEE, 2018.

[6] H.-S. Choi, H. Heo, J. H. Lee, and K. Lee, "Phase-aware single-stage speech denoising and dereverberation with u-net," *arXiv preprint arXiv:2006.00687*, 2020.

[7] J.-M. Valin, "A hybrid dsp/deep learning approach to real-time full-band speech enhancement," in *2018 IEEE 20th international workshop on multimedia signal processing (MMSP)*, pp. 1–5, IEEE, 2018.