A Real-time Sound Source Separation System

Cheng-Yuan Tsai[1], Ting-Yu Lai[2], Chao-Hsiang Hung[2], Hau-Shiang Jung[2], Syu-Siang Wang[2] and Shih-Hau Fang[2,3,*]

[1]*Forensic Science Division, Ministry of Justice Investigation Bureau, New Taipei City 231, Taiwan (R.O.C.)*
[2]*Department of Electrical Engineering, Yuan Ze University, No. 135, Yuandong Rd., Zhongli Dist., Taoyuan City 320, Taiwan (R.O.C.)*
[3]*Department of Electrical Engineering, National Taiwan Normal University, No. 162, Sec. 1, Heping E. Rd., Da'an Dist., Taipei City 106, Taiwan (R.O.C.)*

Abstract
The COVID-19 pandemic has accelerated the shift of many meetings to online platforms. In crucial multi-person meetings, real-time speech separation can significantly enhance subsequent applications. Additionally, real-time speech separation in various speech-related products, such as hearing aids, can improve backend services. However, achieving real-time speech separation poses substantial challenges. It requires segmenting speech over time, and as the information within each segment decreases, the difficulty of source separation increases, along with the necessary computational time. Most previous research has focused on non-real-time speech source separation, and real-time systems often rely on additional information or resources, primarily utilizing English corpora as the dataset. This research aims to develop a real-time speech source separation system. We use the architecture proposed in [1] as our main framework and modify it to handle segmented speech processing. The system is designed to perform real-time computations using a single-core CPU. The scale-invariant signal-to-noise ratio (SI-SNR) is employed as the evaluation metric for objective assessment. We validate the performance of our method with a small dataset in a real-time system and simulate the speech segmentation of the real-time system offline using a large dataset.

Keywords
Speaker Separation, Source Separation, SI-SNR, real-time system

1. Introduction

Sound is vital for human perception and communication, carrying useful information such as scene, speaker, and language. It has many applications like hearing aids, speaker recognition, and speech recognition. Therefore, computer processing of speech signals is a popular research topic.

Source separation is a field that many researchers have delved into with abundant resources, but the signal processing methods used vary. The traditional techniques mainly include beamforming [1, 2] and blind source separation [3], both multi-channel source separation methods. Beamforming uses multiple microphones for omnidirectional reception, determines the direction of the source based on the time of arrival and intensity differences of the signal, and suppresses

The 6th International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT 2024), August 19-22, 2024, Kusatsu, Japan
*Corresponding author.
m55050@mjib.gov.tw (C. Tsai); s1094628@mail.yzu.edu.tw (T. Lai); s1108501@mail.yzu.edu.tw (C. Hung); s1114635@mail.yzu.edu.tw (H. Jung); sypdbhee@saturn.yzu.edu.tw (S. Wang); shfang@saturn.yzu.edu.tw (S. Fang)

all sounds except the target source direction, thereby improving the signal-to-noise ratio (SNR) of the entire system and achieving separation. Blind source separation assumes that each source has independent statistical characteristics and designs filters to separate an unknown number of sources [4]. However, multi-channel signal processing means increased cost. In addition to traditional methods, previous studies have shown that deep learning can achieve excellent separation results in single-channel source separation. Common source separation systems use frequency domain features as inputs, such as in [5, 6, 7, 8], where speech is first converted from time domain to frequency domain signals. However, this process has several drawbacks. Firstly, the conversion process will add a lot of computational cost, and secondly, there is a risk of distortion in the phase and magnitude of the signal during the conversion process. In response to these issues, many scientists have begun to research source separation of time-domain signals, such as in the paper [9], which uses a time-domain audio separation network to overcome the drawbacks of frequency-domain transformation. In the time-domain computations, an encoder-decoder[10, 11, 12, 13] framework is used to model the time-domain signal directly. This approach eliminates the frequency decomposition step and simplifies the separation problem to calculate the output of the encoder, which is then synthesized and restored into the time-domain signal by the decoder. However, this method still results in a large model size and is not suitable for modeling data with longer time series.

Combining the research from both fields, more and more researchers hope to use time-domain signals for source separation. The paper [14] presents a deep learning framework for end-to-end[15, 16, 17, 18, 19] time-domain source separation. It uses a linear encoder to generate optimized speech waveforms for separating each source. The encoder's output is then passed through a set of weight masks estimate[20, 21, 22, 23] to achieve source separation. The separated signals are then reversed back to speech waveforms using a linear decoder. Although this method solves the long delay time required for computing spectrograms and reduces the size of some models, it is still not suitable for processing data with longer time series.

The dual-path recurrent neural network (DPRNN) method in [24] models longer time-series data effectively by splitting the input sequence into smaller chunks. Using the Wall Street Journal dataset (WSJ0-2mix), DPRNN achieved a source-to-interference ratio (SI-SNR) of 18.8 dB in clean environments and 8.4 dB in noisy environments. However, the non-real-time nature and high computational resources required during testing are limitations.

To achieve real-time source separation, methods like multi-input-multi-output (MIMO) require spatial information, increasing costs and resources. Real-time tracking using upper-body humanoid robots and distributed processing, as seen in [25], and fusion methods of camera and microphone array sensors, as in [26], have been explored.

Building on [24], we have enhanced robustness in noisy environments, adapted the system for Chinese corpus, and developed a real-time separation system.

2. MATERIALS AND METHODS

2.1. Database Description

For the evaluation task, we utilized the Taiwan Mandarin Chinese version of the Hearing in Noise Test (TMHINT) [27] to prepare the speech dataset. Each phrase consists of ten characters, encompassing a wide range of commonly used pronunciation types in daily life.

A total of 19 participants, comprising 12 males and 7 females aged between 20 and 28 years, took part in the audio recording for this experiment. None of the participants exhibited noticeable pronunciation difficulties or issues with oral expression, thus representing typical speech conditions.

During the recording process, the equipment, including the laptop array microphone and a standard microphone, was first calibrated for frequency response and to assess the impact of environmental noise on the recording results. The recording took place in a semi-reverberant room measuring 2m x 2m x 2m. The participants sat facing a computer, with the standard microphone positioned at mouth height to minimize distance variation due to differences in participant height. We used Audacity software to control the laptop array microphone and ACQUA software for the standard microphone, both operating at a 48kHz sampling rate. Each participant read the 320 phrases from the TMHINT text sequentially. Each phrase lasted three to four seconds, with a one-second pause between phrases. After reading all 320 phrases, the recordings were stopped in both Audacity and ACQUA, and the files were saved and checked for errors to complete the session.

In the data processing stage, we addressed issues of uneven speech volume and varying speech duration. To ensure consistency, the entire corpus was recorded in one session to avoid variations due to different recording times or changes in speaker conditions. We inserted a blank interval of approximately 1.5 seconds between each sentence to facilitate audio segmentation. We then extracted 320 individual audio files based on these intervals. Voice activity detection (VAD) was performed on the data to remove silent parts, preventing excessive silence in the clips that could negatively impact model training. We manually verified the effectiveness of the VAD method in eliminating silent portions.

2.2. Proposed method

In this paper, we use the DPRNN as our main reference. Figure 1 illustrates the DPRNN process. Initially, an audio clip is input and converted into features that are easily learned by the model through an encoder. The audio then enters the separation model, which is divided into three parts: segmentation processing, DPRNN block processing, and overlap-add.

In segmentation processing, the encoded audio features are broken into block-like segments and connected to form a three-dimensional tensor. Here, L represents the time length, N is the feature-length, W is the input signal, and P is the length of one block. A 50% overlap rate was used in this study. S represents the number of blocks generated during this process, resulting in a $2P * S * N$ three-dimensional tensor.

After obtaining the three-dimensional tensor, DPRNN block processing is performed, which is divided into Intra-chunk RNN and Inter-chunk RNN. The Intra-chunk RNN processes local information within each block in parallel, while the Inter-chunk RNN processes global information across all blocks. Data in the Intra-chunk RNN undergoes a Bi-LSTM, a fully connected layer, and a normalization layer. The Inter-chunk RNN follows the same procedure. These two components combine to form a complete DPRNN block, which can be stacked to increase model depth. In our experiments, we used six stacking layers.

The purpose of overlap-add is to convert the data processed by the DPRNN block back to

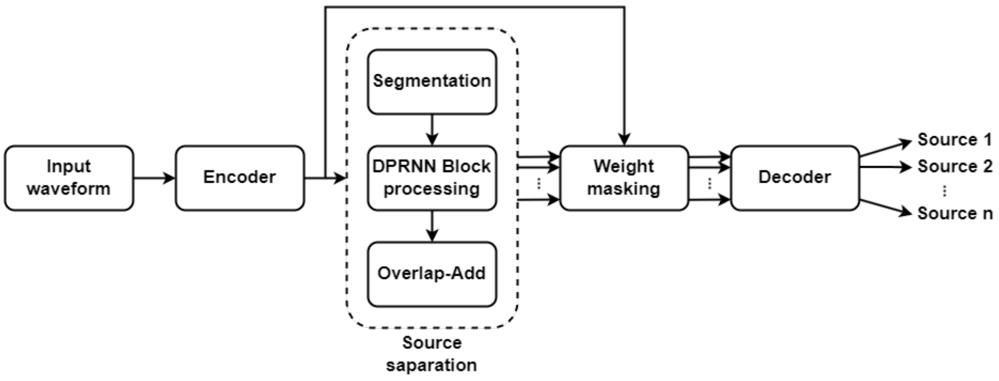


Figure 1: DPRNN block diagram

the input format. As shown in Fig 1, the three-dimensional output of the last DPRNN block is transformed back to a sequence output through overlap-add.

For generating training data, we randomly selected ten males and six females from the previously recorded data, totaling sixteen participants. The training data had a sampling rate of 8kHz. The data from these participants were mixed and divided into three combinations: male-male, male-female, and female-female, with 13,500, 10,000, and 6,500 sentences, respectively. The generation of each combination followed these steps: (1) Two different people were selected from the sixteen, each with 170 sentences of speech signal. (2) One sentence was randomly selected from each person's 170 sentences, resulting in two speech signals. (3) A signal-to-signal ratio between $-5dB$ to $5dB$ was randomly selected, and the two speech signals were mixed to obtain a mixed speech signal. (4) Steps (1) to (3) were repeated to obtain a training corpus of approximately 30 hours, serving as a small training database according to the quantity of each combination.

The test set included four trained speakers and four untrained speakers, each with two males and two females. The test data generation followed these steps: (1) Three gender combinations were used for mixing: male-male, male-female, and female-female. (2) Two speakers were selected within each gender combination, each with 150 speech signals, different from the training set. (3) One sentence was selected from each speaker's 150 speech signals, resulting in two speech signals for each gender combination. (4) The two selected speech signals were mixed at a signal-to-signal energy ratio of 0.5 to obtain a mixed speech signal. (5) Steps (2) to (4) were repeated until each gender combination contained 2000 sentences, resulting in a small test set of 6000 sentences. The number of training and testing data is listed in Table Table 1.

Table 1
Number of training data and testing data.

Data	M-M	M-F	F-F	Total
Training	13500	10000	6500	30000
Testing	2000	2000	2000	6000

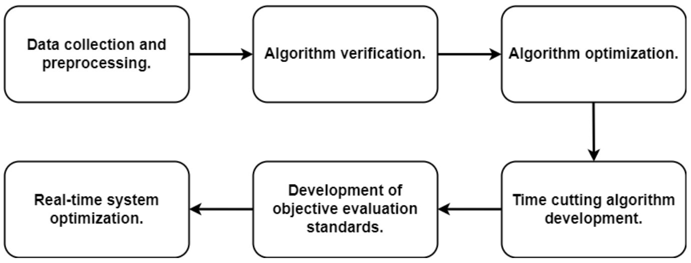

Data collection and preprocessing.
Algorithm verification.
Algorithm optimization.
Time cutting algorithm development.
Development of objective evaluation standards.
Real-time system optimization.

Figure 2: The block diagram of experiment.

3. Real-time System

In this paper, we initially used a mobile phone to play the audio, which we then recorded directly using the computer's microphone. However, this method resulted in the system capturing audio with doubled environmental noise and potentially uneven volume due to human factors during the recording process. To address these issues, we decided to use the mobile phone as the input device for the real-time system, connecting it to the computer via an audio cable. This setup allowed us to avoid the aforementioned problems. In recording mode, the system saves the entire separated audio file to the computer's hard drive at the end of the execution. In playback mode, after separating a segment of the audio, the system plays it in real time through the speakers.

As shown in Fig 2, the experimental steps are as follows: (1) Data collection and preprocessing of audio files. (2) Simulation and computer validation of source separation algorithms. (3) Optimization of algorithm performance. (4) Development of time-domain source separation techniques. (5) Development of real-time system simulation and objective evaluation standards. (6) Real-time system performance optimization.

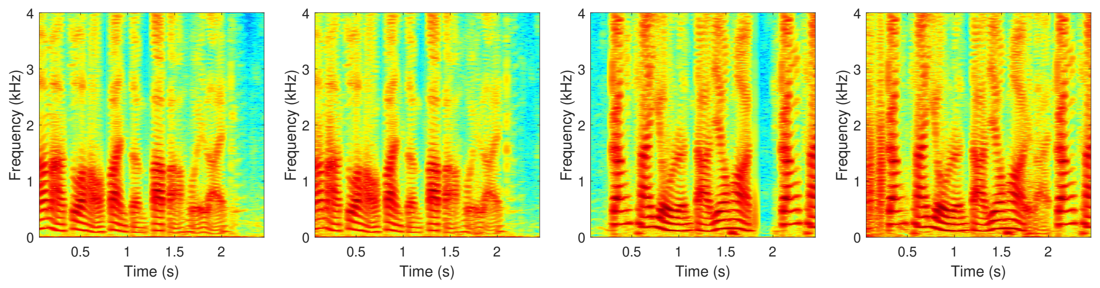
3.1. Recording Mode and Playing Mode

To effectively evaluate the performance of the system, we further expanded it with two modes: recording mode and playback mode. In the recording mode, the system executes the complete process and generates audio files, which are then saved to the computer's hard drive. In the playback mode, the system is capable of playing the separated audio in real-time, allowing users to assess its performance immediately. The application of these two modes enables a more comprehensive evaluation of the system's performance while providing users with a better experience.

3.2. Improved DPRNN Model

To meet the requirements of real-time computation, we cannot input the complete audio file into the model for processing and output the results. Therefore, the audio files need to be segmented, resulting in a reduction in the amount of data input to the model. This reduction in data volume significantly affects the separation performance of the model.

Therefore, we made some improvements to the DPRNN method. We studied the model



(a) Original model speaker1. (b) Improved model speaker1. (c) Original model speaker2. (d) Improved model speaker2.

Figure 3: The results from two different models, "Original model" and "Imprived model".

architecture of DPRNN, where the original DPRNN model first segments an input audio file into several chunks and then stacks them sequentially. The x-axis represents the length of each segmented audio file, and the y-axis represents the number of segmented audio files. The stacked segmented audio files are then fed into the RNN block of the DPRNN, which consists of two parts: intra-chunk operations based on the temporal sequence of the inputs and inter-chunk operations across the segmented audio files. In the context of real-time computation, the audio is segmented before each model inference, resulting in short segmented audio files being processed each time. As a result, the effectiveness of the inter-chunk RNN (Inter chunk) that operates across time on the segmented audio files is significantly reduced. In situations where inter-chunk RNN does not provide significant benefits, we choose to omit it. This saves half of the computational workload and reduces the impact of audio file segmentation on the system. Finally, considering model performance and computational time trade-offs, we settled on a model architecture with 4 layers of intra-chunk RNN.

In addition to the 4 layers of intra-chunk RNN, we believe that even though the inter-chunk RNN may not provide substantial help when the audio files are segmented and stacked, having a single layer of inter-chunk RNN can still aid the model in feature learning. Therefore, based on the architecture of 4 layers of intra-chunk RNN, while maintaining its computational workload, we replace the last layer of intra-chunk RNN with inter-chunk RNN. The final architecture consists of 3 layers of intra-chunk RNN plus 1 layer of inter-chunk RNN.

The original full model, consisting of 6 layers, had a size of 31MB. We adjusted the model architecture to 3 layers of intra-chunk plus 1 layer of inter-chunk, achieving real-time computational requirements, and reducing the model size to approximately 10MB.

Fig 3 shows the spectrograms of the results before and after model adjustment. It can be observed that there is little impact on the results before and after model adjustment, and the spectrograms of both still exhibit a high degree of similarity.

3.3. Computer hardware and software connections in the system

In the system, it is necessary to establish connections between the model and external audio sources. To achieve this purpose, we utilize a tool called "pyaudio". Pyaudio is an open-source Python tool developed by a third party, providing capabilities for recording, reading audio files, and playing audio files within Python. The use of this tool enables us to easily apply audio sources and transmit audio data to our system for further processing.

One of the advantages of pyaudio is its cross-platform compatibility. It supports various operating systems, including Linux, Microsoft Windows, and Apple macOS, making our system theoretically capable of functioning across different operating system environments.

With pyaudio, we can effortlessly set up recording or playback functions for audio sources and connect audio data to our model through appropriate configurations. This facilitates an effective linkage between audio source input and model separation, equipping the system with essential tools and functionalities for smooth operation.

Overall, the utilization of pyaudio allows us to seamlessly handle audio within the Python environment and connect it to our model. This empowers our system with enhanced capabilities and flexibility, enabling us to achieve the goals of efficient and accurate audio separation and real-time playback.

Furthermore, while using Pyaudio, it's important to note that the precision of audio data input is in 16-bit format, whereas the model's computation requires data in Float32 format. Hence, before feeding data into the model for computation, a conversion from 16-bit to Float32 format is necessary to ensure proper data processing by the model. Once the model's computation is complete, the results must be converted back to a 16-bit format for audio data storage using pyaudio.

4. Results

4.1. SI-SNR

The Scale-Invariant Signal-to-Noise Ratio (SI-SNR)[28], is a metric used to evaluate the performance of signal separation. SI-SNR measures the ratio between the separated signal and the original mixed signal, and it effectively quantifies the separation quality between the signal and the noise. SI-SNR is defined as the following equation 1:

$$\begin{cases} S_{target} := \frac{\langle \hat{s}, s \rangle s}{\|s\|^2} \\ e_{noise} := \hat{s} - s_{target} \\ SI - SNR := 10lg\frac{\|s_{target}\|^2}{\|e_{noise}\|^2} \end{cases} \tag{1}$$

4.2. Dual-channel engineering method

Since calculating SI-SNR requires aligning every point of the two audio signals, there may be human errors in real-time systems, such as from the start of the system to playing the audio file, or from the end of the audio file to the end of the system, which can lead to imprecise separation of the audio files. Therefore, we propose a dual-channel engineering method to address this issue. Fig 4 shows the core of this method. That is the dual-channel approach that simultaneously mixes the target audio file with the mixed audio file and changes the system's input to the processed dual-channel audio file. This ensures that the blank spaces in the output audio file and the target audio file are of the same length, thereby achieving alignment.

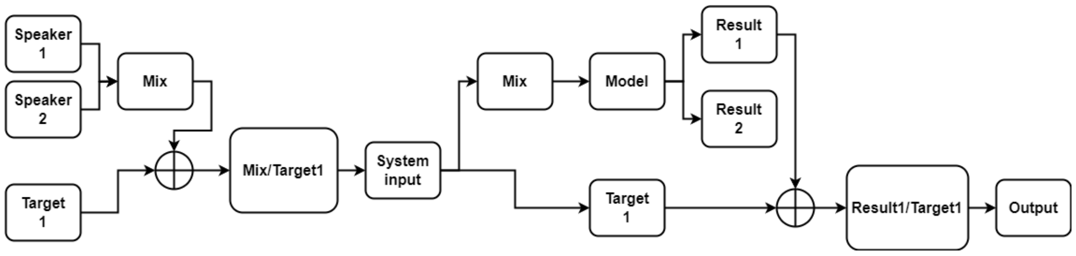


Figure 4: The block diagram of Dual-channel engineering met.

The reason for doing this is to align the timing of the mixed speech and the target speech. At the same time, we want the input and output of the system in recording mode to be a dual-channel audio file. Therefore, we first merge the mixed speech and target speech A into a dual-channel audio file, where the first channel is the mixed speech and the second channel is the target speech A. This ensures that both enter the system simultaneously and are output simultaneously, with aligned timing. However, once inside the system, only the first channel, which is the mixed speech, enters the model for separation. After going through the model, we obtain a separate speech A, while the target speech A does not enter the model. Before outputting, we mix these two speech segments back into a dual-channel audio file.

4.3. Result of the Models in Clean and Noisy Speech

In Table 2, we conducted experiments using the audio files mentioned earlier, analyzing male-male, male-female, and female-female combinations. The DPRNN model achieved an average SI-SNR=16.7dB, with male-female pairs performing the best at SI-SNR=17.9 dB. We speculate that this is due to the greater difference in frequency range between male and female voices, making it easier for the model to separate the two.

Additionally, we added two types of noise, computer fan noise, and pink noise, to the training and testing data used in a clean background environment. The noise was mixed into the data at SNRs of 0, 5, and 10. We randomly selected one type of noise and one SNR and mixed it into 30,000 sentences of training data and 6,000 sentences of testing data. We trained the model with the noisy data, changing the output from two speakers to three, including two speakers and the noise. The purpose of this was to treat the noise as a third speaker for separation.

We tested the model on audio files in both noisy and clean background environments, with an average SI-SNR of 14.9 in the noisy environment. Male-male, male-female, and female-female pairs achieved SI-SNRs of 15.5, 15.8, and 13.2, respectively.

Table 2
Comparing different models under various combinations with SI-SNR(dB) in a clean background environment. M represents male, and F represents female.

Data	M-M	M-F	F-F	Average
clean	17.5	17.9	14.7	16.7
noisy	15.5	15.8	13.2	14.9

4.4. Result of Real-time System

Real-time System Performance in a Small Dataset: We selected five sentences from the clean test data to conduct small data testing on a single-core CPU. The selection criterion was based on an average SNR of 16.7. We chose two sentences above the average SNR, two sentences below the average, and one sentence approximately equal to the average, resulting in a total of five sentences. As shown in Table 3, we tested the impact of different input sizes on the system's performance. The input sizes were 800 and 2000 points per input. It can be observed that the smaller the input size, the poorer the system's performance. This is because our main method is based on LSTM, which is highly influenced by the length of the input sequence. If the information available for each model processing is reduced, its performance will be adversely affected.

4.5. Result of the Offline Simulation in Large Dataset

Due to the limitation of obtaining only one audio segment at a time in the real-time system, evaluating the performance with a large amount of data would be time-consuming. Therefore, we conducted offline simulations by manually segmenting the audio files to mimic the segmentation approach in the real-time system. These segmented audio files were then processed through the model in separate batches, and the resulting segmented audio segments were concatenated to form a complete sentence. We compared five different models: a 6-layer full model, a 4-layer full model, a 2-layer full model, a 4-layer intra-chunk model, and a 3-layer intra-chunk with a 1-layer inter-chunk model, running on a single-core CPU. We tested the input lengths of 800 samples and 2000 samples. Table 4 shows that the adjusted models outperformed the original DPRNN model significantly. Compared to the 6-layer full model, the 4-layer intra-chunk model achieved a 0.74 dB improvement in SI-SNR for the input length of 2000 samples, and a decrease of 0.15 dB in SI-SNR for the input length of 800 samples. It also showed improvements compared to other full models. In the 3-layer intra-chunk with the 1-layer inter-chunk model, there was a 2.28 dB improvement in SI-SNR for the input length of 2000 samples compared to the 4-layer intra-chunk model, and improvements compared to other models for the input length of 800 samples. Therefore, we concluded that when the audio is segmented in time and the input length is shorter, adding a significant number of inter-chunk layers would lead to a decrease in model performance. However, not adding any inter-chunk layer would also not achieve the best performance. Thus, the 3-layer intra-chunk with 1-layer inter-chunk model exhibited the

Table 3
The SI-SNR(dB) values of each sentence in the real-time system.

Sentence	Seg=800	Seg=2000
I	10.9	16.1
II	1.23	4.2
III	-0.52	4.97
IV	14.9	16.39
V	10.9	13.37
Average	7.48	**11.0**

best performance in both input length scenarios.

Table 4
SI-SNR(dB) of different models at different input sizes.

Model	Seg=800	Seg=2000
6 layers full	5.09	8.36
4 layers full	5.16	8.12
2 layers full	4.05	6.43
4 layers intra	4.94	9.1
3 layers intra and 1 layer inter	**6.88**	**11.38**

5. Conclusion

This study focuses on developing a real-time speech source separation system in a single-core CPU computing environment. First, we validate the performance of the complete DPRNN model on Chinese speech data and in the presence of noise backgrounds, demonstrating its feasibility. Next, we modify the DPRNN model architecture to meet the requirements of a real-time system. In a real-time system, the input audio needs to be segmented over time. Therefore, we change the original model's structure to a 3-layer intra-chunk plus 1-layer inter-chunk architecture, which satisfies the computational time constraints of a real-time system using a single-core CPU. The objective evaluation criterion used is SI-SNR. The 3-layer intra-chunk plus 1-layer inter-chunk model architecture performs best even with an input length of 2000 samples. Additionally, we develop a dual-channel engineering approach to evaluate the performance of the real-time system under a small dataset scenario. Again, the 3-layer intra-chunk plus 1-layer inter-chunk model architecture achieves the best performance evaluation results with an input length of 2000 samples. Furthermore, the 3-layer intra-chunk plus 1-layer inter-chunk model architecture meets the real-time computational time requirements in both single-core CPU and GPU computations.

However, the current performance is maintained when the input length is 2000 samples, which corresponds to a minimum of 250 milliseconds (ms) processing time. In applications that require audiovisual synchronization, this delay is noticeable to users. Therefore, future research can focus on achieving a balance between performance and reducing the input length per model inference, aiming to minimize latency.

6. Acknowledgments

The authors would like to thank the financial support from the Ministry of Justice (113-1301-05-04-01) and the National Science and Technology Council (NSTC112-2221-E-155-017-MY2 and NSTC112-2222-E-155-002).

References

[1] Z.-Q. Wang, H. Erdogan, S. Wisdom, K. Wilson, D. Raj, S. Watanabe, Z. Chen, J. R. Hershey, Sequential Multi-Frame neural beamforming for speech separation and enhancement, in: Spoken Language Technology Workshop (SLT), IEEE, 2021, pp. 905–911.

[2] M. Z. Ikram, D. R. Morgan, A beamforming approach to permutation alignment for multichannel frequency-domain blind speech separation, in: International Conference on Acoustics, Speech, and Signal Processing, volume 1, IEEE, 2002, pp. I–881–I–884.

[3] Y. Benabderrahmane, S. A. Selouani, D. O'Shaughnessy, Blind speech separation for convolutive mixtures using an oriented principal components analysis method, in: 18th European Signal Processing Conference, 2010, pp. 1553–1557.

[4] A. Liutkus, J.-L. Durrieu, L. Daudet, G. Richard, An overview of informed audio source separation, in: 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), 2013, pp. 1–4.

[5] T. Virtanen, Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria, IEEE Transactions on Audio, Speech, and Language Processing 15 (2007) 1066–1074.

[6] T. Hasumi, T. Kobayashi, T. Ogawa, Investigation of network architecture for single-channel end-to-end denoising, in: 28th European Signal Processing Conference (EUSIPCO), 2021, pp. 441–445.

[7] R. J. Issa, Y. F. Al-Irhaym, Audio source separation using supervised deep neural network, in: Journal of Physics: Conference Series, volume 1879, IOP Publishing, 2021, pp. 1–8.

[8] R. Watanabe, D. Kitamura, H. Saruwatari, Y. Takahashi, K. Kondo, DNN-Based frequency component prediction for frequency-domain audio source separation, in: 28th European Signal Processing Conference (EUSIPCO), 2021, pp. 805–809.

[9] Y. Luo, N. Mesgarani, TaSNet: Time-Domain audio separation network for real-time, single-channel speech separation, in: International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 696–700.

[10] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2017) 2481–2495.

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[12] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[13] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, LSTM-based encoder-decoder for multi-sensor anomaly detection, arXiv preprint arXiv:1607.00148 (2016).

[14] Y. Luo, N. Mesgarani, Conv-Tasnet: Surpassing ideal time–frequency magnitude masking for speech separation, IEEE/ACM Transactions on Audio, Speech, and Language Processing 27 (2019) 1256–1266.

[15] S.-W. Fu, T.-W. Wang, Y. Tsao, X. Lu, H. Kawai, End-to-end waveform utterance enhance-

ment for direct evaluation metrics optimization by fully Convolutional Neural Networks, IEEE/ACM Transactions on Audio, Speech, and Language Processing 26 (2018) 1570–1584.

[16] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, E. Dupoux, End-to-end speech recognition from the raw waveform, arXiv preprint arXiv:1806.07098 (2018).

[17] F. Lluís, J. Pons, X. Serra, End-to-end music source separation: Is it possible in the waveform domain?, arXiv preprint arXiv:1810.12187 (2018).

[18] Y. Luo, Z. Chen, N. Mesgarani, T. Yoshioka, End-to-end microphone permutation and number invariant multi-channel speech separation, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 6394–6398.

[19] Z.-Q. Wang, J. L. Roux, D. Wang, J. R. Hershey, End-to-end speech separation with unfolded iterative phase reconstruction, arXiv preprint arXiv:1804.10204 (2018).

[20] Y. Isik, J. L. Roux, Z. Chen, S. Watanabe, J. R. Hershey, Single-channel multi-speaker separation using deep clustering, arXiv preprint arXiv:1607.02173 (2016).

[21] D. Yu, M. Kolbæk, Z.-H. Tan, J. Jensen, Permutation invariant training of deep models for speaker-independent multi-talker speech separation, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 241–245.

[22] M. Kolbæk, D. Yu, Z.-H. Tan, J. Jensen, Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks, IEEE/ACM Transactions on Audio, Speech, and Language Processing 25 (2017) 1901–1913.

[23] Y. Luo, Z. Chen, N. Mesgarani, Speaker-independent speech separation with deep attractor network, IEEE/ACM Transactions on Audio, Speech, and Language Processing 26 (2018) 787–796.

[24] Y. Luo, Z. Chen, T. Yoshioka, Dual-Path RNN: Efficient long sequence modeling for time-domain single-channel speech separation, in: International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 46–50.

[25] K. Nakadai, K. Hidai, H. Okuno, H. Kitano, Real-time speaker localization and speech separation by audio-visual integration, in: Proceedings 2002 IEEE International Conference on Robotics and Automation, volume 1, 2002, pp. 1043–1049 vol.1.

[26] C.-F. Liu, W.-S. Ciou, P.-T. Chen, Y.-C. Du, A real-time speech separation method based on camera and microphone array sensors fusion approach, Sensors 20 (2020).

[27] M. Huang, Development of taiwan mandarin hearing in noise test, Department of speech language pathology and audiology, National Taipei University of Nursing and Health science (2005).

[28] J. Le Roux, S. Wisdom, H. Erdogan, J. R. Hershey, SDR–half-baked or well done?, in: International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 626–630.