INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

A Simple RNN Model for Lightweight, Low-compute and Low-latency Multichannel Speech Enhancement in the Time Domain

Ashutosh Pandey[1], *Ke Tan*[1], *Buye Xu*[1]

[1]Reality Labs Research at Meta

apandey620@meta.com, tanke1116@meta.com, xub@meta.com

Abstract

Deep learning has led to unprecedented advances in speech enhancement. However, deep neural networks (DNNs) typically require large amount of computation, memory, signal buffer and processing time to achieve strong performance. Designing a DNN to meet a given resource constraint requires dedicated efforts. This study proposes a novel recurrent neural network (RNN) based model for time-domain multichannel speech enhancement that can be easily tuned to meet a given constraint. We present results of training the model at different scales, where algorithmic latency varies from 1 ms to 16 ms, model size varies from 100 Thousand to 25 Million parameters, and compute to process one second of speech varies from 100 Mega to 25 Giga multiply-accumulates (MACs). Experimental results demonstrate that the proposed model can obtain similar or better performance using fewer computes and parameters than competitive approaches to low-latency multichannel speech enhancement.

Index Terms: Multichannel speech enhancement, time-domain speech enhancement, RNN, lightweight, low-compute, low-latency

1. Introduction

Acoustic interference, such as background noise and room reverberation, is unavoidable in the real world. It decreases the intelligibility and quality of speech, creating problems for humans and machines. Speech enhancement is concerned with removing acoustic interference from speech signals. Monaural speech enhancement uses recordings from a single microphone and can utilize only spectral information. In contrast, multichannel speech enhancement uses multiple microphones to exploit additional spatial information for improved discrimination between target and interference [1].

Multichannel speech enhancement using deep neural networks has been extensively investigated in the past few years. One of the popular approaches is to combine a DNN with a traditional spatial filter, such as a mask-based MVDR beamformer where the DNN is used to estimate the speech and noise statistics for the spatial filter [2, 3, 1]. Another approach is to train DNNs with input features encoding explicit spatial information [4, 5]. A more recent trend is to use complex spectrum mapping [6] for complex spectrogram enhancement [4, 7, 8, 9, 10, 11] or waveform mapping [12] for time-domain enhancement [13, 14, 15, 16, 17]. Although end-to-end training has led to dramatic improvements in the performance, traditional spatial filters are still widely utilized even with stronger models [4, 18].

While DNNs obtain impressive enhancement, they come with a major drawback; they require enormous amount of resources, such as processing time, computation, run-time memory, and power consumption. The processing time consists of algorithmic and computational latency. Existing studies have attempted to reduce the algorithmic latency by reducing the window size in short-time processing [15, 19, 18]. Computational latency, on the other hand, is dependent on the amount of computation and hardware implementation. In research, a popular approach to reducing computational latency has been to reduce the total amount of computation, which also leads to reduction in the power consumption [20, 21].

The run-time memory is a function of the model size and the dependency graph of computation. The model size is measured by the number of parameters in the network, and it has been a standard practice to report parameter efficiency to advocate for an efficient model [22, 23]. The dependency graph of computation, however, determines how computations are performed within a network, and has been largely ignored in the past studies. One common case is that a given layer in the network uses outputs from many of the previous layers or the past outputs, which leads to a higher run-time memory. For example, popular approaches use UNet-based [18, 24] or fully-convolutional networks [25, 15, 19] with skip connections and dilated convolutions with large temporal contexts. However, they may not be ideal for devices requiring small run-time memory. Other cases include heavy use of convolutional layers and attention, where weight sharing and attention map computation could lead to high memory consumption even though the model size is relatively small.

Optimizing a DNN model for speech enhancement with a given resource constraint can be a daunting task, since DNNs require a lot of tuning of hyperparameters to achieve good performance. Therefore, there is a need to develop a model or framework that provides efficient mechanism for controlling different aspects of resource constraints while providing satisfactory speech enhancement. This study attempts to fill this gap by proposing a simple recurrent neural network (RNN) model for multichannel speech enhancement in the time domain. The model takes as input the raw samples of noisy speech, performs denoising and dereverberation, and outputs raw samples of enhanced speech.

The time-domain approach is adopted to provide the model with a capability to utilize end-to-end training for learning features suited for the particular task of speech enhancement with a given constraint. Moreover, a spatial processing block, inspired by frequency-dependent spatial filtering, is proposed for efficient spatial context aggregation. The motivation to use RNNs is two-fold. First, RNNs only require a run-time memory proportional to the size of the hidden state. In contrast to the dilated convolutions [25, 24, 26] and self-attention [27, 28] for temporal context aggregation, RNNs summarize the past infor-

10.21437/Interspeech.2023-2418

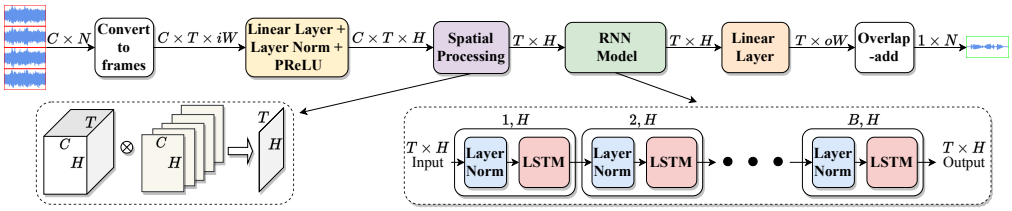

Figure 1: *The proposed multichannel RNN model.*

mation in a single hidden vector. Consequently, the output of a given time frame depends only on the previous hidden state and the current input. Second, RNNs are highly effective for robust speech enhancement in challenging conditions [29, 30, 27].

The proposed model uses one hyperparameter to manipulate algorithmic latency from 1 ms to 16 ms. Similarly, the number of parameters and computations are controlled by one hyperparameter. We train models with number of parameters varying from 100 Thousand (K) to 25 Million. The compute to process one second of audio is varied from 100 Mega to 25 Giga multiply-accumulates (MACs). Additionally, the model is designed to have minimal run-time memory as it comprises only a stack of several RNN layers.

2. Problem Definition

Given an array with C microphones, a multichannel recording $\boldsymbol{Y} \in \mathbb{R}^{C \times N}$ is defined as

$$\boldsymbol{Y} = \boldsymbol{S}_d + \boldsymbol{S}_R + \boldsymbol{N} \tag{1}$$

where N is the number of samples, $\boldsymbol{S}_d$, $\boldsymbol{S}_R$ and $\boldsymbol{N} \in \mathbb{R}^{C \times N}$ and respectively represent direct-path speech, its reverberation, and interfering signals arriving at microphones. The goal of multichannel speech enhancement is to get a good estimate of the direct-path speech $\boldsymbol{s}_{dr}$ at a reference microphone r using noisy recording $\boldsymbol{Y}$. In other words, the objective is to remove room reverberation and noises from the degraded speech at a reference microphone.

3. RNN for Multichannel Speech Enhancement

The proposed RNN model for multichannel speech enhancement is shown in Fig. 1. A given multichannel input signal of shape $C \times N$ is converted to overlapping frames with an input window size of iW and a hop size of S, leading to a tensor of shape $C \times T \times iW$ with T frames. Next, all the frames are projected to a higher dimension of size H using a linear layer followed by layer normalization [31] and parametric rectified linear unit (PReLU) nonlinearity [32].

After this, the tensor of size $C \times T \times H$ is reduced along the channel dimension using a spatial processing block. The spatial processing block comprises H trainable filters of length C that are applied across channels. Such filtering resembles frequency-dependent filtering in traditional spatial signal processing with H corresponding to the number of frequency bins. These time-invariant filters are independent of the microphone signals at inference time, unlike traditional adaptive spatial filters. The output of the spatial processing block encodes spatial, spectral and temporal information to be utilized by the following modules in the network.

The spatial processing block is followed by an RNN model comprising a stack of B long short-term memory (LSTM) blocks. Each LSTM block consists of a layer normalization followed by one layer of LSTM with hidden size H. The use of LSTM ensures that the output for a given input frame is dependent only on the the input in the current frame and the hidden states computed from one past frame. The output of the RNN model is projected to the output window size of oW using a linear layer. Finally, overlap-add is applied to obtain the enhanced signal at the reference microphone.

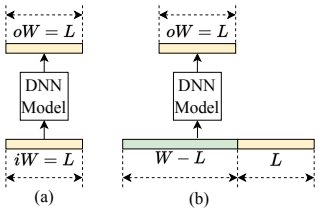

Figure 2: *Explored approaches to achieve a desired latency L. (a) Minimum input context, (b) Fixed input context of length $W > L$.*

The hyperparameter H represents the width of the network and is used for manipulating the number of computation and parameters. Adopting LSTM as the recurrent layer topology leads to quadratic relation between H and the number of computations and parameters. In other words, doubling the value of H quadruples the number of computation and parameters. This relation can be changed to linear by employing the idea of group LSTMs with two groups [33, 34]. We decided not to investigate this because of the significantly increased training time due to the double number of LSTMs and consistently subpar performance with similar width. We also explored other RNNs, such as gated recurrent unit (GRU) and simple recurrent unit (SRU), but decided to stick with LSTM because of its superior performance.

3.1. Algorithmic Latency

The latency of a speech enhancement system comprises of algorithmic latency and computational latency. The computational latency is dependent on the amount of compute and hardware implementation, and the algorithmic latency is attributed to the employed algorithm. For a system using short-time processing, the algorithmic latency is equal to the employed window size. In this work, we modify iW and oW to obtain a a desired algorithmic latency of L while keeping the hop size S fixed. We investigate the following two approaches.

3.1.1. Enhancement with Minimum Input Context

This approach, shown in Fig. 2(a), uses a value of L for both iW and oW to achieve a latency of L. We call this approach *minimum window context*, as this is the minimum input context

Table 1: *Performance of the proposed model for $H = 256$ varying latencies using two approaches: (a) minimum input context, b) fixed input context of* 16 ms.

	STOI (%)						PESQ						SNR					
Mix.	58.8						1.42						-6.6					
$C \rightarrow$	2		4		8		2		4		8		2		4		8	
$L \downarrow$	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
1 ms	72.8	73.7	76.9	78.0	80.4	80.8	1.86	1.91	2.00	2.06	2.14	2.15	3.3	3.4	4.2	4.4	4.8	4.9
2 ms	74.0	74.9	78.4	78.8	81.3	81.7	1.92	1.95	2.07	2.11	2.19	2.21	3.4	3.6	4.5	4.6	5.0	5.2
4 ms	74.6	75.5	78.9	79.5	82.0	82.3	1.93	1.97	2.09	2.12	2.21	2.23	3.6	3.8	4.6	4.8	5.2	5.2
8 ms	76.3	76.7	80.0	80.3	83.1	83.0	2.03	2.04	2.17	2.20	2.29	2.30	4.0	4.1	5.0	5.0	5.5	5.6
16 ms	77.8	77.8	80.5	80.5	83.1	83.1	2.13	2.13	2.25	2.25	2.36	2.36	4.2	4.2	5.2	5.2	5.7	5.7

Table 2: *Performance of the proposed model for an algorithmic latency of* 2 ms *with varying widths.*

		STOI (%)			PESQ			SNR			MACs			Params		
Mix.		58.8			1.42			-6.6								
$H \downarrow$	$C \rightarrow$	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8
64	(a)	66.2	68.2	70.2	1.64	1.68	1.75	1.9	2.2	2.6	108.42M	113.02M	122.24M	104.67K	104.80K	105.06K
	(b)	66.1	68.9	71.4	1.63	1.70	1.82	1.9	2.3	2.9	139.01M	172.75M	240.24M	119.01K	119.14K	119.40K
128	(a)	70.1	73.8	77.2	1.76	1.87	2.02	2.7	3.4	3.9	413.44M	422.66M	441.09M	405.92K	406.18K	406.70K
	(b)	71.2	74.6	77.3	1.80	1.92	2.02	2.8	3.5	4.0	477.37M	544.87M	679.85M	434.59K	434.85K	435.37K
256	(a)	74.0	78.4	81.3	1.92	2.07	2.19	3.4	4.5	5.0	1.61G	1.63G	1.67G	1.60M	1.60M	1.60M
	(b)	74.9	78.8	81.7	1.95	2.11	2.21	3.6	4.6	5.2	1.75G	1.89G	2.16G	1.66M	1.66M	1.66M
512	(a)	77.1	81.6	84.4	2.03	2.23	2.34	4.3	5.3	6.0	6.37G	6.41G	6.48G	6.34M	6.34M	6.35M
	(b)	77.9	82.1	85.0	2.07	2.24	2.39	4.4	5.5	6.1	6.69G	6.96G	7.50G	6.46M	6.46M	6.46M
1024	(a)	79.1	83.9	86.6	2.13	2.34	2.48	4.7	6.0	6.7	25.33G	25.40G	25.55G	25.27M	25.27M	25.27M
	(b)	80.1	84.2	87.0	2.18	2.37	2.52	5.0	6.2	6.9	26.15G	26.69G	27.77G	25.50M	25.50M	25.50M

that can be used to achieve a latency of L. Assuming that we want to exploit overlap-add, this approach can provide an algorithmic latency of L = $2 \times S$. We use $S = 1$ ms, which amounts to an algorithmic latency of 2 ms. To achieve a latency of 1 ms, we use $iW = 2$ ms, $oW = 2$ ms, and pad the input with 1 ms of zeros in the beginning, which forces the network to predict one frame ahead of time.

3.1.2. Enhancement with Fixed Input Context

In this approach, shown in Fig. 2(b), iW is set to a fixed value of $W > L$ and oW is set to L. The input signal is padded with zeros of length $W - L$ in the beginning to ensure that the output frame corresponds to the last oW samples in the input frame. A similar approach of dual-window short-time Fourier transform (STFT) has been used in [18] for low-latency complex spectral mapping. We train models for L from $\{1, 2, 4, 8, 16\}$ ms. The W is set to 16 ms. The oW is set to L for all the cases except for 1 ms, where it is set to 2 ms to exploit overlap-add, and the input is padded with zeros of length $W - L + 1$ in the beginning.

4. Experiments and Results

4.1. Dataset

We use the Interspeech2020 DNS Challenge corpus [35] to generate pairs of clean and noisy signals. All the speakers from the training set are randomly split into pairs of training, test and validation speakers using a ratio of 85 %, 5 % and 10 %. Similarly, noises are partitioned into distinct sets of training, test and validation. An eight-microphone circular array with a radius of 10 cm is used to create multichannel data. We use a data generation algorithm described in past studies [16, 9, 17] to generate 10 seconds long 80K training, 1.6K validation, and 3.2K test utterances.

The generated dataset can be be considered as a challenging dataset for speech enhancement. The T60 for reverberation is uniformly sampled from $[0.2, 1.3]$ seconds. The target speech is contaminated by 5 to 10 noise sources, its own reverberation, and reverberation of noise sources. The signal-to-noise (SNR) between the direct-path speech and interferences (excluding speech reverberation) is sampled uniformly from $[-10, 10]$ dB. Moreover, the training target is set to be the direct-path speech at the first microphone ($r = 1$), implying that the model is trained for the difficult task of end-to-end joint denoising and dereverberation.

4.2. Experimental Settings

All the utterances are resampled to 16 kHz before data generation. We use signals from the 1^{st} and 5^{th} microphones for 2-channel models, from 1^{st}, 3^{rd}, 5^{th}, and 7^{th} microphones for 4-channel models, and from all the microphones for 8-channel models. A given mutichannel waveform input to a model is normalized (multiplied with a scalar) to have an overall variance of one across the microphones.

All the models are developed, trained and evaluated using PyTorch. We set the number of LSTM blocks (B) to 3. We train models for 100 epochs using random chunks of 4 seconds cropped out of 10 seconds long utterances with a batch size of 16. The phase constrained magnitude (PCM) loss proposed in [26] is used for training all the models. The Adam optimizer [36] with $amsgrad = True$ and a constant learning rate of 0.0002 is used. The gradient norm is clipped to a value of 0.03. The combination of gradient clipping and $amsgrad = True$ is found to be critical to stabilize the training of LSTMs with large sequences. Additionally, a combination of automatic mixed precision (AMP) and Nvidia V100 GPUs is utilized for a much faster training.

Table 3: *Performance comparison with baseline models. RNN-H-(a) represents a model using approach (a) with width H.*

		STOI (%)			PESQ			SNR			MACs (Giga)			Params (Millions)		
		58.8			1.42			-6.6								
$L\downarrow$	Model $\downarrow, C\rightarrow$	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8
4 ms	MC-CRN [18]	**77.2**	79.8	81.5	**2.05**	2.14	2.20	3.9	4.7	4.9	3.5	3.5	3.6	2.32	2.32	2.32
	RNN-300-(a)	75.5	80.0	83.1	1.96	2.15	**2.28**	3.8	4.9	**5.5**	**2.2**	**2.3**	**2.4**	**2.21**	**2.21**	**2.21**
	RNN-300-(b)	76.3	**80.3**	**83.2**	2.01	**2.17**	**2.28**	**4.0**	**5.0**	**5.5**	2.4	2.5	2.9	2.27	2.27	2.27
	RNN-1024-(b)	81.3	85.1	87.6	2.23	2.42	2.54	5.4	6.5	7.1	26.1	26.7	27.8	25.53	25.53	25.54
2 ms	MC-CRN [18]	**75.8**	78.6	80.1	**1.99**	2.06	2.14	3.6	4.4	4.6	3.5	3.5	3.6	2.32	2.32	2.32
	MC-Conv-TasNet [15]	75.5	79.1	81.8	**1.99**	2.14	2.25	3.6	4.6	5.2	5.0	5.0	5.1	5.04	5.07	5.13
	RNN-300-(a)	74.8	79.0	82.3	1.95	2.11	2.23	3.8	4.7	5.2	**2.2**	**2.2**	**2.3**	**2.19**	**2.19**	**2.19**
	RNN-300-(b)	75.4	**79.7**	**82.9**	1.96	**2.15**	**2.27**	**3.8**	**4.8**	**5.4**	2.4	2.5	2.9	2.26	2.26	2.26
	RNN-1024-(b)	80.1	84.2	87.0	2.18	2.37	2.52	5.0	6.2	6.9	26.2	26.7	27.8	25.50	25.50	25.50
1 ms	RNN-300-(a)	73.8	78.2	81.3	1.91	2.07	2.18	3.4	4.4	5.0	**2.2**	**2.2**	**2.3**	**2.19**	**2.19**	**2.19**
	RNN-300-(b)	**74.5**	**79.0**	**81.9**	**1.92**	**2.11**	**2.22**	**3.6**	**4.6**	**5.1**	2.4	2.5	2.9	2.26	2.26	2.26
	RNN-1024-(b)	79.3	83.3	87.0	2.14	2.31	2.51	4.9	5.9	6.9	26.2	26.7	27.8	25.50	25.50	25.50

All the models are evaluated using short-time objective intelligibility (STOI) [37], perceptual evaluation of speech quality (PESQ) [38] and SNR. The direct-path speech at the first microphone is used as the reference to compute all the metrics. Average scores over 3.2K test utterances are reported. The amount of computation is reported in MAC for processing one second of speech. The $Thop$[1] library in Python is utilized for MAC computation.

4.3. Baseline Models

We compare the proposed system with existing approaches to low-latency multichannel speech enhancement. First, we train a multichannel convolutional time-domain audio separation network (MC-Conv-TasNet) proposed in [15, 19]. The best-performing configuration from [25] with a filter length of 32 and stride of 16 is used for separation network. A convolutional layer using filters of length $C \times 32$ with 512 features followed by layer normalization is used to obtain spatial representation, and it is concatenated with the encoder output of the reference channel before the separator network. The MC-ConvTasNet is a time-domain model with a latency of 2 ms.

Secondly, we train a recently proposed multichannel convolutional recurrent network (MC-CRN) for complex spectral mapping [18]. The network utilizes a dual-window approach to achieve a small latency of 4 ms. It also uses future frame prediction to achieve a latency of 2 ms without much drop in the performance.

4.4. Results

First, we discuss the performance of RNN model, reported in Table 1, when latency is varied from 1 ms to 16 ms. In all the tables, (a) refers to the minimum input context approach (Section 3.1.1) and (b) refers to the fixed window approach with an input context of 16 ms (Section 3.1.2). We observe that a larger input context obtains consistent improvements over minimum input context. Additionally, consistent and significant improvements are obtained with a larger latency for both (a) and (b). For example, STOI scores are respectively improved by 4.1, 2.5, and 2.3 for 2-channel, 4-channel and 8-channel models when latency is increased from 1 ms to 16 ms using approach (b). In summary, we are able to provide a latency vs performance trade-off by changing just one parameter L that decides the values of iW and oW (see Section 3.1) for both (a) and (b).

[1] `https://pypi.org/project/thop/`

Next, we provide a performance vs number of computation and parameters trade-off by fixing the algorithmic latency to 2 ms and varying H from a small value of 64 to a large value of 1024. Results are given in Table 2. We notice that the smallest model ($H = 64$) takes around 120 Mega (M) MACs to process one second of speech with 8 channels. It can improve the STOI by 11.4, PESQ by 0.33 and SNR by 9.6 by using only 105K parameters. We also notice that (b) obtains consistent improvement over (a) for all H with a slight increase in the number of computations and parameters. The biggest 8-channel model with $H = 1024$ can obtain impressive improvements of 28.8 in STOI, 1.1 in PESQ and 13.8 in SNR with a small algorithmic latency of 2 ms by using a compute of 25G with 25M parameters. The results in Table 1 and Table 2 illustrate that the proposed framework is general enough to be employed for a wide range of resource constraints.

Finally, we compare the proposed model with baseline models in Table 3. We use $H = 300$ to make the number of parameters comparable to that of MC-CRN. For the latency of 4 ms, RNN-300-(b) model obtains superior performance than MC-CRN for 4-channel and 8-channel models while using a much smaller amount of computations. It is similar in SNR and PESQ but slightly worse in STOI for the 2-channel case. For the latency of 2 ms, RNN-300-(b) models is similar or better than the stronger baseline model MC-Conv-TasNet, while using much smaller number of parameters and computations. We also notice that RNN-300-(b) with a latency 1 ms matches to the performance of MC-Conv-TasNet with latency 2 ms for the case of 4 and 8 channels. We also present result for RNN-1024-(b) to highlight that the performance can be significantly improved even for smaller latencies, such as 1 ms and 2 ms, by increasing the number of parameters and computations.

5. Conclusions

We have proposed a simple RNN model for the task of joint denoising and dereverberation in the time domain. The model can operate with an ultra small algorithmic latency and is also efficient in terms of run-time memory and computation. It is competitive with existing approaches to low-latency speech enhancement while using a much smaller amount of computation and parameters. Additionally, we have trained it for a wide range of resource constraints, demonstrating that it is suitable for multiple applications with different resource requirements.

6. References

[1] S. Gannot, E. Vincent, S. Markovich-Golan, and A. Ozerov, "A consolidated perspective on multimicrophone speech enhancement and source separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 692–730, 2017.

[2] H. Erdogan, J. R. Hershey, S. Watanabe, M. I. Mandel, and J. Le Roux, "Improved MVDR beamforming using single-channel mask prediction networks," in *INTERSPEECH*, 2016, pp. 1981–1985.

[3] J. Heymann, L. Drude, and R. Haeb-Umbach, "Neural network based spectral mask estimation for acoustic beamforming," in *ICASSP*, 2016, pp. 196–200.

[4] Z.-Q. Wang, J. Le Roux, and J. R. Hershey, "Multi-channel deep clustering: Discriminative spectral and spatial embeddings for speaker-independent speech separation," in *ICASSP*, 2018, pp. 1–5.

[5] Z.-Q. Wang and D. Wang, "Combining spectral and spatial features for deep learning based blind speaker separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 457–468, 2018.

[6] S.-W. Fu, T.-y. Hu, Y. Tsao, and X. Lu, "Complex spectrogram enhancement by convolutional neural network with multi-metrics learning," in *Workshop on Machine Learning for Signal Processing*, 2017, pp. 1–6.

[7] B. Tolooshams, R. Giri, A. H. Song, U. Isik, and A. Krishnaswamy, "Channel-attention dense U-Net for multichannel speech enhancement," in *ICASSP*, 2020, pp. 836–840.

[8] K. Tan, Z.-Q. Wang, and D. Wang, "Neural spectrospatial filtering," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 605–621, 2022.

[9] A. Pandey, B. Xu, A. Kumar, J. Donley, P. Calamia, and D. L. Wang, "Multichannel speech enhancement without beamforming," in *ICASSP*, 2022, pp. 6502–6506.

[10] J. Liu and X. Zhang, "DRC-NET: Densely connected recurrent convolutional neural network for speech dereverberation," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 166–170.

[11] D. Lee and J.-W. Choi, "DeFT-AN: Dense frequency-time attentive network for multichannel speech enhancement," *IEEE Signal Processing Letters*, vol. 30, pp. 155–159, 2023.

[12] S. Fu, Y. Tsao, X. Lu, and H. Kawai, "Raw waveform-based speech enhancement by fully convolutional networks," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2017, pp. 006–012.

[13] C.-L. Liu, S.-W. Fu, Y.-J. Li, J.-W. Huang, H.-M. Wang, and Y. Tsao, "Multichannel speech enhancement by raw waveform-mapping using fully convolutional networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1888–1900, 2020.

[14] Y. Luo, Z. Chen, N. Mesgarani, and T. Yoshioka, "End-to-end microphone permutation and number invariant multi-channel speech separation," in *ICASSP*, 2020, pp. 6394–6398.

[15] J. Zhang, C. Zorilă, R. Doddipatla, and J. Barker, "On end-to-end multi-channel time domain speech separation in reverberant environments," in *ICASSP*, 2020, pp. 6389–6393.

[16] A. Pandey, B. Xu, A. Kumar, J. Donley, P. Calamia, and D. L. Wang, "TPARN: Triple-path attentive recurrent network for time-domain multichannel speech enhancement," in *ICASSP*, 2022, pp. 6497–6501.

[17] ——, "Time-domain ad-hoc array speech enhancement using a triple-path network," in *INTERSPEECH*, 2022, pp. 729–733.

[18] Z.-Q. Wang, G. Wichern, S. Watanabe, and J. Le Roux, "STFT-domain neural speech enhancement with very low algorithmic latency," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 397–410, 2022.

[19] Z. Tu, J. Zhang, N. Ma, J. Barker *et al.*, "A two-stage end-to-end system for speech-in-noise hearing aid processing," *Proc. Clarity*, pp. 3–5, 2021.

[20] C. Haruta and N. Ono, "A low-computational dnn-based speech enhancement for hearing aids based on element selection," in *European Signal Processing Conference*, 2021, pp. 1025–1029.

[21] K. Tan and D. Wang, "Towards model compression for deep learning based speech enhancement," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 29, pp. 1785–1794, 2021.

[22] L. Lee, Y. Ji, M. Lee, M.-S. Choi, and N. Coporation, "DEMUCS-Mobile: On-device lightweight speech enhancement." in *Interspeech*, 2021, pp. 2711–2715.

[23] Y. Luo, C. Han, and N. Mesgarani, "Ultra-lightweight speech separation via group communication," in *ICASSP*, 2021, pp. 16–20.

[24] A. Pandey and D. L. Wang, "TCNN: Temporal convolutional neural network for real-time speech enhancement in the time domain," in *ICASSP*, 2019, pp. 6875–6879.

[25] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time-frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1256–1266, 2019.

[26] A. Pandey and D. Wang, "Dense CNN with self-attention for time-domain speech enhancement," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1270–1279, 2021.

[27] ——, "Self-attending RNN for speech enhancement to improve cross-corpus generalization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1374–1385, 2022.

[28] H. Zhang, A. Pandey, and D. L. Wang, "Low-latency active noise control using attentive recurrent network," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1114–1123, 2023.

[29] A. Pandey and D. L. Wang, "On cross-corpus generalization of deep learning based speech enhancement," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, in press, 2020.

[30] ——, "Learning complex spectral mapping for speech enhancement with improved cross-corpus generalization," in *INTERSPEECH*, 2020, p. in press.

[31] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv:1607.06450*, 2016.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015, pp. 1026–1034.

[33] F. Gao, L. Wu, L. Zhao, T. Qin, X. Cheng, and T.-Y. Liu, "Efficient sequence learning with group recurrent networks," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 799–808.

[34] K. Tan and D. L. Wang, "Learning complex spectral mapping with gated convolutional recurrent networks for monaural speech enhancement," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 380–390, 2019.

[35] C. K. Reddy, V. Gopal, R. Cutler, E. Beyrami, R. Cheng, H. Dubey, S. Matusevych, R. Aichner, A. Aazami, S. Braun *et al.*, "The INTERSPEECH 2020 deep noise suppression challenge: Datasets, subjective testing framework, and challenge results," in *INTERSPEECH*, pp. 2492–2496.

[36] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[37] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 2125–2136, 2011.

[38] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ) - a new method for speech quality assessment of telephone networks and codecs," in *ICASSP*, 2001, pp. 749–752.