INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

DEMUCS-Mobile : On-device lightweight speech enhancement

Lukas Lee, Youna Ji, Minjae Lee, Min-Seok Choi

Naver Coporation, South Korea
{lukas.lee, youna.ji, mjlee.0328, minseok.choi}@navercorp.com

Abstract

As the importance of speech enhancement for real-world application increases, the compactness of the model is also becoming a crucial study. In this paper, we present compression techniques to reduce the model size and applied them to the state-of-the-art real-time speech enhancement system. We successfully reduce the model size by actively applying channel pruning while maintaining performance. In particular, we propose a method to prune more channels of convolutional neural networks (CNN) by utilizing gated linear unit (GLU) activation. In addition, lower-bit-quantization is applied to reduce model size, while minimizing performance degradation caused by quantization. We show the performance of our proposed model on a mobile device where computing resources are limited. In particular, it is implemented to enable streaming, and speech enhancement works in real-time.

Index Terms: speech enhancement, on-device, mobile, channel pruning, model compression

1. Introduction

Speech enhancement is widely applied for achieving robust speech communication in real-world applications. It is used to separate the clean target speech from a noise-corrupted input to enhance the clarity of noisy signals. Since deep learning was introduced to speech enhancement as part of prior studies, remarkable performance improvements in terms of quality and intelligibility have been yielded [1, 2].

The general approach of these studies has been to take a short-time Fourier transform (STFT) representation of the noisy signal as an input and estimate the ideal Time-Frequency mask or spectral mapping between noisy and clean spectra [3, 4]. Although this approach shows promising results, it requires the STFT which involves a computational expense and causes speech distortion because of phase inconsistency, particularly in a low signal-to-noise ratio (SNR) environment [5].

To overcome the aforementioned limitations, a number of recent works propose a deep neural network (DNN) model in the waveform domain [6, 7, 8, 9, 10]. In this approach, the DNN model directly generates a de-noising speech signal, so it does not require additional computation such as an STFT, and does not suffer from phase in-consistency. In [6], the SEGAN used an end-to-end convolutional neural network so that feature extraction and separation can be implicitly incorporated into the network architecture. The Wave-U-Net [7, 11] is a one-dimensional U-Net with a convolutional encoder-decoder with skip connection. In [12], the fully-convolutional time-domain audio separation network (Conv-TasNet) is introduced. More recently, DEMUCS architecture was introduced in [13, 14] for audio separation and speech enhancement. These studies show the effectiveness of the convolutional encoder-decoder architecture for time-domain speech enhancement.

Meanwhile, the demand for on-device speech processing is increasing [15, 16, 17, 18]. On-device processing secures user privacy and is free from communication latency. Despite recent development in speech enhancement and the need for on-device processing, on-device implementation is still a challenging issue. Most works are focusing on speech enhancement performance and introduce networks that require large memory and high computational cost.

Considering the necessity of developing a lightweight speech enhancement model, we reduced the size of the convolutional neural network (CNN) based models with considerably less performance degradation. In particular, using the study [13, 14] as a baseline, we verified the performance of the lightweight speech enhancement model. First of all, we employ the channel pruning of CNN, which was introduced in the field of computer vision [19, 20]. Especially, we propose a new method that prunes the CNN channels beside the gated linear unit (GLU). We trained the neural network in a way to decay less important channels and analyzed the number of channels that were reduced for each layer.

Quantization is employed as well to further reduce the size of the neural network. We found that 16-bit quantization generates a clean waveform without the performance loss compared to the 32-bit floating point model.

We run the lightweight model solely on a mobile device, without communication with a server. In addition, we develop a streaming inference mode in smartphones. When tested on smartphones, our model run in real-time, within 0.7 Real-time factor (RTF, e.g. time to enhance a frame divided by the stride [14]).

2. CNN based speech enhancement

In this paper, we used DEMUCS architecture described in [14] as a baseline since this research showed state-of-the-art level performance. In addition, it can be implemented for real-time systems so it is suitable as a starting point for building an on-device model.

The DEMUCS architecture is initially presented in [13] for music source separation and adopted to real-time speech enhancement in [14]. It consists of a convolutional encoder-decoder network with a U-Net skip connection and LSTM is applied after the encoder. In this structure, an encoder takes raw audio of noisy signal as an input and produces a latent representation. Then uni-directional LSTM is followed for sequence modeling. In addition, each encoder and decoder layer uses a GLU activation [21], so it reduces the output dimension by half.

3. On-device speech enhancement

3.1. Channel pruning by inducing sparsity

Convolutional encoder-decoder is a commonly used architecture in the speech enhancement field [6, 7, 14]. In this paper, we present compression methods that can be effectively applied

Algorithm 1 The modified ADAM optimizer update scheme for sparse-inducing training

```
$\theta_\mathbf{t}$: Model parameter at step $t$
$\mathbf{m_t}$ and $\mathbf{v_t}$: First- and second-moment vector at step $t$
$\mathbf{g_t}$: Computed gradients at step $t$
$\alpha$: Learning rate
$\beta_1, \beta_2$: The exponential decay rate for the 1st/2nd moment
$\gamma$: $BatchNorm$ scaling factor
$\mathbf{BN_{decay}}$: $BatchNorm$ decay rate

if $\theta_\mathbf{t} \in \{\gamma\}$ then
    $\mathbf{g_t} \leftarrow \nabla_\theta \mathbf{f_t}(\theta_{t-1}) + \mathbf{sign}(\theta_\mathbf{t}) \cdot \mathbf{BN_{decay}}$
else
    $\mathbf{g_t} \leftarrow \nabla_\theta \mathbf{f_t}(\theta_{t-1})$
end if
$\mathbf{m_t} \leftarrow \beta_1 \cdot \mathbf{m_{t-1}} + (1-\beta_1) \cdot \mathbf{g_t}$
$\mathbf{v_t} \leftarrow \beta_2 \cdot \mathbf{v_{t-1}} + (1-\beta_2) \cdot \mathbf{g_t^2}$
$\widehat{\mathbf{m_t}} \leftarrow \frac{\mathbf{m_t}}{1-\beta_1^t}, \widehat{\mathbf{v_t}} \leftarrow \frac{\mathbf{v_t}}{1-\beta_2^t}$
$\widehat{\theta_t} \leftarrow \theta_{t-1} - \alpha \cdot \frac{\widehat{\mathbf{m_t}}}{\sqrt{\widehat{\mathbf{v_t}}} + \epsilon}$
```

to those architectures.

Channel pruning has been suggested as a method to reduce the CNN model size in the area of computer vision [22, 23]. Among the channel pruning methods, a method of decaying the scaling factor of the batch normalization layer has also been studied [19, 20]. During the training stage, the scaling factors of the batch normalization layers are lowered for purposes. The L1 loss term is used to make the scaling factors sparse during the training stage. According to [19, 20], sparse-inducing training works well in the image classification with SGD optimizer.

After the scaling factors become sparse, a channel of the CNN layer connected to the decayed scaling factor of the batch normalization layer can be deleted as the channel is supposed to be multiplied by the decayed one.

We implemented the sparse-inducing training in ADAM optimizer, which is shown in Algorithm 1. By modifying the iteration scheme of ADAM optimizer, scaling factors of the batch normalization layers are decayed.

3.2. Network design for channel pruning

We change the DEMUCS structure to prune the networks efficiently. Batch normalization layers are inserted into the initial encoders and decoders of DEMUCS structure in intended positions. Each unit of a encoder and a decoder now contains CNN-activation-batch normalization-CNN(or transposed CNN)-activation, from input to output.

In this pattern, scaling factors of one batch normalization layer can decide the sparsity of double CNN layers. Each output channel of the first CNN and each input channel of the second CNN (or transposed CNN) are connected independently to other channel information. Therefore, when one scaling factor decays, the corresponding one output/input channel of a previous/next CNN layer can be pruned at once, resulting in a double-pruning effect. This procedure is shown in Figure 1.

3.3. Pruning method for a GLU

However, this design for channel pruning can only be applied when an activation function such as ReLU is used, which has

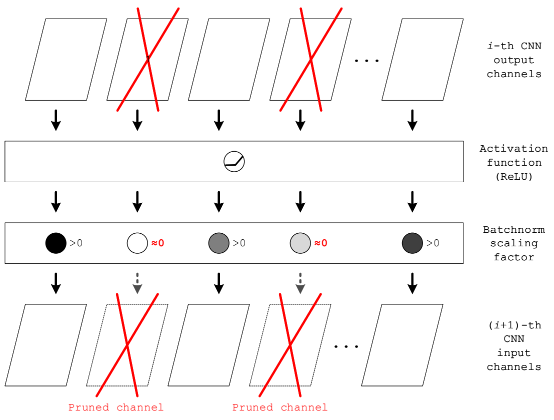

Figure 1: *The encoder/decoder structure with batch normalization scaling factor for efficient pruning.*

the same input and output dimensions, and the output of each dimension does not depend on the other input dimension's operations. With GLU activation, the design in 3.2 does not work anymore. The usage of GLU activation here is as follows.

$$\mathbf{h}^{l+1}(\mathbf{X}) = \mathbf{h}^l_\mathbf{L}(\mathbf{X}) \odot \sigma(\mathbf{h}^l_\mathbf{R}(\mathbf{X})), \tag{1}$$

where $\mathbf{X}$ denotes the input of the neural network and $\mathbf{h}^l(\mathbf{X})$ represents the hidden units of layer l. Splitting the tensor $\mathbf{h}^l(\mathbf{X})$ into half, with respect to hidden dimensions, $\mathbf{h}^l_\mathbf{L}(\mathbf{X})$ and $\mathbf{h}^l_\mathbf{R}(\mathbf{X})$ of layer l are generated. And $\odot$ is element-wise multiplication and σ denotes the sigmoid function.

With GLU operations, one dimension in the output of the activation function depends on the other dimensions in the activation input. To perform pruning with GLU activation, we suggest a new method to bypass such limits and maximize channel pruning.

Our proposed method utilizes the sparsity of hidden vectors and batch normalization scaling factors. Denoting the decayed batch normalization scaling factor as "zero" (whose value is under a specific threshold value), the pruning process with GLU is conducted as follows.

1. Set the initial sparsity "zero"s, after sparse-inducing training.
2. Split the dimensions (d : the length of the hidden dimensions) of the batch normalization layer into left and right.
3. Ignore the initial "zero"s on the right side.
4. Copy the "zero"s on the left side to the right side. The distance between the "zero"s on the left side and the copied "zero"s on the right side is $d/2$.
5. Set the standard of pruning CNN channels as (1) "zero"s on the left side and (2) "zero"s on the right side which are only shifted from the left side.
6. Prune corresponding CNN channels with new "zero"s.

Figure 2 explains this algorithm.

3.4. Quantization

Concerning the on-device usage in a smartphone, we apply lower bit quantization from the 32-bit floating point. Tensorflow-lite is used as a tool to efficiently carry out lower-bit operations. In particular, we employ 16-bit quantization to further reduce model size.

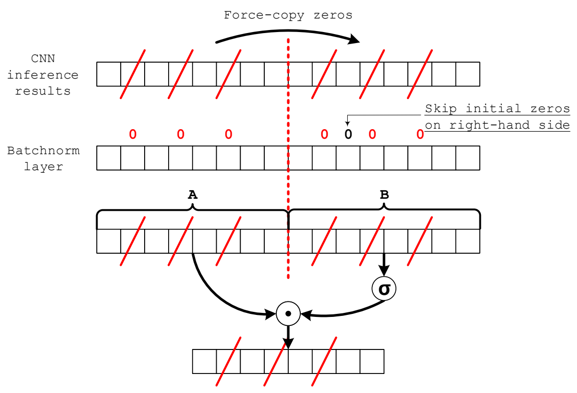

Figure 2: *The algorithm of copying the sparsity with GLU*

3.5. On-device inference

We put the lightweight model on a real smartphone to verify the possibility of a practical on-device inference. Inference relies only on the hardware of a smartphone without a support of a server resources. In particular, it is implemented to enable streaming on a smartphone to verify the inference time in an actual streaming environment.

In a mobile device, streaming speech enhancement is performed with a frame window and overlapping stride. Like in [14], upsampling the input frame at the beginning and downsampling to the network output is conducted every frame we process. Extra frames were employed for better resampling.

For an efficient operation, we concatenated an input frame and an extra frame so that those two can be computed in neural networks at once. As extra frames are just for better resampling, the information of a input frame and extra frame should be separated afterwards. Proper zero-padding was added to separate the target frame after the computation.

4. Experiments

4.1. Experimental setups and baselines

We trained the baseline 32-bit floating point model with these hyperparameters; a number of layers L=5, the initial number of hidden channel H=48, layer kernel size K=8, stride S=4, and resampling factor U=4. The modified ADAM optimizer described in Algorithm 1 with an initial learning rate of 3.e-4 was utilized. For training and evaluation, Valentini et al. [24] was used and it consists of 28 speakers, and speech corrupted by 10 types of noises at 4 levels of SNR: 15 dB, 10 dB, 5 dB, 0 dB. Originally the dataset was recorded at 48 kHz and we resampled it to 16 kHz. Furthermore, the L1 loss for the waveform domain and multi-resolution STFT loss [14, 25, 26] for the spectral domain were applied with a weight of 0.5.

Regarding data augmentation, we applied three methods as described in [14].

- Remix - It shuffles the noises to generate new mixtures.
- Bandmask – it is a band-stop filter to apply the same effect of SpecAug [27] augmentation in the time-domain signal.
- Random shift – the signal is randomly shifted between 0 to S second and we set S to 0.5 sec.

For evaluation, we used wide-band perceptual evaluation of speech quality (PESQ) [28] and the short-time objective intelligibility (STOI) [29] which are commonly used evaluation matrices for speech enhancement. The PESQ aims to predict the subjective quality with a value from 0.5 to 4.5 and the STOI aims to predict the intelligibility with a value from 0 to 100. The testset of Valentini benchmark [24] was used to measure PESQ and STOI.

For streaming evaluation on a mobile device, the speech enhancement model processes a 40 ms frame at once and a stride of 16 ms is taken, so that input waveforms can be enhanced in a overlapping manner.

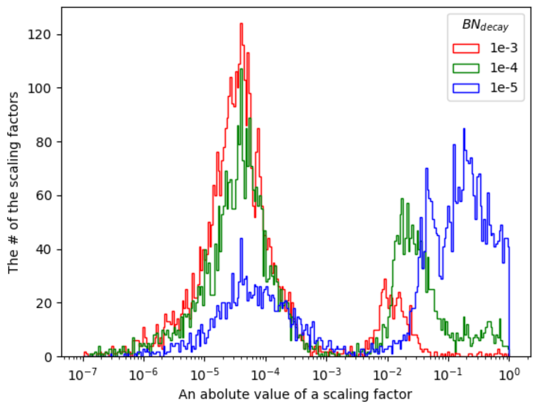

Figure 3: *Distribution of batch normalization scaling factor, according to the BatchNorm decay rate during the training ($\mathbf{BN_{decay}}$), after training over 400 epochs on Valentini dataset.*

Table 1: *Results of sparse inducing training on scaling factor. BN_{decay} is the BatchNorm decay rate during training.*

Model	Model size	PESQ	STOI
Noisy	-	1.97	91.5
Non-streamable			
SEGAN [6]	-	2.16	-
Wave U-Net [11]	-	2.40	-
MetricGAN [30]	-	2.86	-
Demucs-Large [14]	128 MB	3.07	95
streamable			
DeepMMSE [31]	-	2.77	93
Demucs [14]	73 MB	2.93	95
streamable, sparse-inducing training			
BN_{decay}=0	73 MB	2.89	94.5
BN_{decay}=1e-5	73 MB	2.87	94.7
BN_{decay}=1e-4	73 MB	2.90	94.3
BN_{decay}=1e-3	73 MB	2.76	93.9

4.2. Experimental results

In this section, we present the evaluation results of the baseline and the proposed pruning model. Regarding the notation,

- λ : Threshold value for pruning. Every scaling factors, whose absolute value is under the value of λ, is decayed.
- p : $\frac{\text{the \# of decayed scaling factors}}{\text{The total \# of scaling factors}}$. Pruning ratio.
- E_i, D_i : The i-th encoder and decoder respectively, i=1, 2, .., 5

Sparse-inducing training. At the initial stage, the BatchNorm decay rate $\mathbf{BN_{decay}}$ was set to 1e-3,1e-4 and 1e-5 and the performance was checked to determine the decay rate.

Figure 3 shows the distribution of scaling factors after 400 epochs of training for each BatchNorm decay rate. The higher decay rate, the more scaling factor tends to decay. This means that there are more layers to be able to prune. On the other hand, Table 1 shows that performance degradation occurs when the largest batch norm decay rate is used. It means the scaling factors are more decayed for the network to function properly. Therefore, we set the BatchNorm decay rate as 1e-4, in which case little performance degradation is caused while scaling factors decay considerably.

To analyze the effect of sparse induction training, we draw the degree of sparsity on each encoder and decoder layer as training progresses. In Figure 4, the degree of brightness in the heatmap denotes the degree of sparsity in each batch normalization layer by every training epoch.

The scaling factors decayed more near the end of the encoder and the beginning of the decoder during training, as shown in Figure 4. It is the result from sparse-inducing training with $\mathbf{BN_{decay}}$=1e-4, which shows little performance degradation. This means that the CNN channels adjacent to latent representation can be pruned more with reduced performance loss. Further, the LSTM layers are located between the encoder and decoder, and we assumed that a smaller LSTM dimension is required to guarantee performance, following the distribution in Figure 4. Accordingly, we aggressively reduced the dimensions of the two-layer LSTM from 768 to 250.

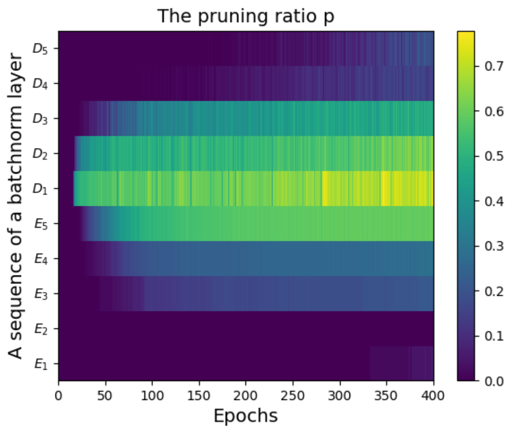

Figure 4: *Over 400 epochs' training, distribution of pruning ratio p in each batch normalization layer, given threshold λ=1e-4 and BN_{decay} of 1e-4 . Higher ratio (higher brightness) represents higher sparsity.*

Table 2: *Channel pruning results. λ is a pruning threshold value and p is the pruning ratio.*

Model	Model size	LSTM dim.	PESQ	STOI
p=0, No pruning	73.85 MB	768	2.90	94.3
p=0, No pruning	34.61 MB	250	2.89	94.4
p=0.62, λ=6.56e-4	18.25 MB	250	2.89	94.4
p=0.65, λ=1.19e-2	17.28 MB	250	2.74	94.3

Results of channel pruning. Before the pruning process, we obtained a model of approximately 35 MB with sparse-inducing training and two-layers of unidirectional LSTM with 250 dimensions. The BatchNorm decay rate of 1e-4 was chosen.

Table 3: *The channel pruning results in detail. C1 is the first CNN layer of an encoder or decoder module. T.C2 is the transposed layer in a module in the decoder. "Out/In" represents each out/in convolutional channels.*

Layer	channel #	new channel #	Pruned (%)
E_1- C1:Out, C2:In	48, 48	38, 38	20.8
E_2- C1:Out, C2:In	96, 96	94, 94	2.1
E_3- C1:Out, C2:In	192, 192	174, 174	9.4
E_4- C1:Out, C2:In	384, 384	311, 311	19.0
E_5- C1:Out, C2:In	768, 468	356, 356	53.6
D_1- C1:Out, T.C2:In	1536, 768	394, 197	74.3
D_2- C1:Out, T.C2:In	768, 384	324, 162	57.8
D_3- C1:Out, T.C2:In	384, 192	254, 127	33.9
D_4- C1:Out, T.C2:In	192, 96	188, 94	2.1
D_5- C1:Out, T.C2:In	96, 48	78, 39	18.8

Table 2 shows that even the half-sized-model has comparable performance in terms of the PESQ and STOI values. The detailed result of channel pruning is shown in Table 3.

Results of quantization. Additionally, we conducted quantization on the pruned model to further reduce the model size. In Table 4, the result of the 16-bit quantization is shown. The model has no performance degradation with quantization.

Table 4: *Quantization Results*

Model	Model size	PESQ	STOI
Pruned (Float32)	18.25 MB	2.89	94.4
Pruned (Float16)	8.90 MB	2.89	94.4

Results of on-device inference on a smartphone. The RTF results are shown in Table 5. A Samsung Galaxy S21 Ultra smartphone was used to conduct inference tests on the speech enhancement model. Our lightweight model runs within 0.7 RTF in a streaming manner, using the "on-device" hardware.

Table 5: *Real-time factors of the on-device inference measured on a smartphone*

Model	Model size	Type	RTF
**server*			
DEMUCS [14]	73 MB	streaming	0.6
**mobile device*			
Pruned (Float32)	18.25 MB	batch	0.11
Pruned (Float16)	8.90 MB	batch	0.13
**mobile device*			
Pruned (Float32)	18.25 MB	streaming	0.56
Pruned (Float16)	8.90 MB	streaming	0.62

5. Conclusions

In this paper, we present model compression techniques that can effectively apply to convolutional encoder-decoder architecture. We have made the lightweight model for on-device speech enhancement and successfully implemented it on the smartphone. We induced the scaling factors of batch normalization layers to be sparse, which leads CNN-based speech enhance model to be pruned. A new method for maximizing channel pruning with GLU is also suggested. With sparse-inducing training and channel pruning, we compressed the model size for speech enhancement approximately half, maintaining performance. Quantization is further applied to compress the pruned model. Without or little performance degradation, we made the speech enhancement model work under 10 MB, and the lightweight runs in real-time using only on-device hardware.

6. References

[1] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal processing letters*, vol. 21, no. 1, pp. 65–68, 2013.

[2] ——, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.

[3] S. R. Park and J. W. Lee, "A fully convolutional neural network for speech enhancement," *Proc. Interspeech 2017*, pp. 1993–1997, 2017.

[4] K. Wilson, M. Chinen, J. Thorpe, B. Patton, J. Hershey, R. A. Saurous, J. Skoglund, and R. F. Lyon, "Exploring tradeoffs in models for low-latency speech enhancement," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 366–370.

[5] K. Paliwal, K. W'ojcicki, and B. Shannon, "The importance of phase in speech enhancement," *speech communication*, vol. 53, no. 4, pp. 465–494, 2011.

[6] S. Pascual, A. Bonafonte, and J. Serrà, "Segan: Speech enhancement generative adversarial network," *Proc. Interspeech 2017*, pp. 3642–3646, 2017.

[7] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *arXiv preprint arXiv:1806.03185*, 2018.

[8] A. Pandey and D. Wang, "Tcnn: Temporal convolutional neural network for real-time speech enhancement in the time domain," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6875–6879.

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

[10] R. Giri, U. Isik, and A. Krishnaswamy, "Attention wave-u-net for speech enhancement," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 249–253.

[11] C. Macartney and T. Weyde, "Improved speech enhancement with the wave-u-net," *arXiv preprint arXiv:1811.11307*, 2018.

[12] 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*, vol. 27, no. 8, pp. 1256–1266, 2019.

[13] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," *arXiv preprint arXiv:1911.13254*, 2019.

[14] A. Defossez, G. Synnaeve, and Y. Adi, "Real time speech enhancement in the waveform domain," *arXiv preprint arXiv:2006.12847*, 2020.

[15] J.-Y. Wu, C. Yu, S.-W. Fu, C.-T. Liu, S.-Y. Chien, and Y. Tsao, "Increasing compactness of deep learning based speech enhancement models with parameter pruning and quantization techniques," *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1887–1891, 2019.

[16] K. Tan, X. Zhang, and D. Wang, "Real-time speech enhancement using an efficient convolutional recurrent network for dual-microphone mobile phones in close-talk scenarios," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5751–5755.

[17] Q. Wang, I. L. Moreno, M. Saglam, K. Wilson, A. Chiao, R. Liu, Y. He, W. Li, J. Pelecanos, M. Nika *et al.*, "Voicefilter-lite: Streaming targeted voice separation for on-device speech recognition," *Proc. Interspeech 2020*, pp. 2677–2681, 2020.

[18] I. Fedorov, M. Stamenovic, C. Jensen, L.-C. Yang, A. Mandell, Y. Gan, M. Mattina, and P. N. Whatmough, "Tinylstms: Efficient neural speech enhancement for hearing aids," *arXiv preprint arXiv:2005.11138*, 2020.

[19] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.

[20] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *International Conference on Learning Representations*, 2018.

[21] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 121–125.

[22] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[23] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J.-H. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *NeurIPS*, 2018.

[24] C. Valentini-Botinhao *et al.*, "Noisy speech database for training speech enhancement algorithms and tts models," 2017.

[25] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6199–6203.

[26] ——, "Probability density distillation with generative adversarial networks for high-quality parallel waveform generation," *arXiv preprint arXiv:1904.04472*, 2019.

[27] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[28] I.-T. Recommendation, "Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," *Rec. ITU-T P. 862*, 2001.

[29] 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*, vol. 19, no. 7, pp. 2125–2136, 2011.

[30] S.-W. Fu, C.-F. Liao, Y. Tsao, and S.-D. Lin, "Metricgan: Generative adversarial networks based black-box metric scores optimization for speech enhancement," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2031–2041.

[31] Q. Zhang, A. Nicolson, M. Wang, K. K. Paliwal, and C. Wang, "Deepmmse: A deep learning approach to mmse-based noise power spectral density estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1404–1415, 2020.