INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

VoiceFilter-Lite: Streaming Targeted Voice Separation for On-Device Speech Recognition

Quan Wang, Ignacio Lopez Moreno, Mert Saglam, Kevin Wilson, Alan Chiao, Renjie Liu, Yanzhang He, Wei Li, Jason Pelecanos, Marily Nika, Alexander Gruenstein

Google LLC, USA

quanw@google.com

Abstract

We introduce VoiceFilter-Lite, a single-channel source separation model that runs on the device to preserve only the speech signals from a target user, as part of a streaming speech recognition system. Delivering such a model presents numerous challenges: It should improve the performance when the input signal consists of overlapped speech, and must not hurt the speech recognition performance under all other acoustic conditions. Besides, this model must be tiny, fast, and perform inference in a streaming fashion, in order to have minimal impact on CPU, memory, battery and latency. We propose novel techniques to meet these multi-faceted requirements, including using a new asymmetric loss, and adopting adaptive runtime suppression strength. We also show that such a model can be quantized as a 8-bit integer model and run in realtime.

Index Terms: source separation, speaker recognition, speech recognition, asymmetric loss, adaptive suppression

1. Introduction

As deep learning gains its popularity in speech and signal processing, we have seen impressive progress in recent years using blind source separation techniques for multi-talker speech recognition. This is often referred to as the "cocktail-party problem" [1]. Some of the representative advances include deep clustering [2], deep attractor network [3], and permutation invariant training [4, 5].

Voice filtering, also known as *speaker extraction*, is a slightly different problem than blind speech separation. In the voice filtering setup, we aim to separate the speech of a target speaker, whose identity is known. The target speaker could be represented by either a one-hot vector from a closed speaker set [6], or a speaker-discriminative voice embedding, such as an i-vector [7] or d-vector [8]. Several different frameworks have been developed for the task of speaker-conditioned speech separation in recent years, including DENet [9], SpeakerBeam [10], and VoiceFilter [11]. A speaker-conditioned separation system has several advantages over blind separation systems that make it more applicable in real applications: (1) It is not required to know the number of sources, which is usually not available at runtime. The same model works for both single-speaker and multi-speaker scenarios. (2) It does not have the speaker-permutation problem. (3) It does not require an output channel selection step after the separation, which in blind separation systems is usually implemented by selecting the loudest output channel, or employing an additional speaker verification system.

However, integrating a speaker-conditioned speech separation model to production environments, especially on-device automatic speech recognition (ASR) systems such as [12], presents a number of challenges. Quality wise, the model should not only improve the ASR performance when there are multiple voices, but also should be harmless to the recognition performance under other scenarios, *e.g.* when the speech is only from the target speaker, or when there is non-speech noise such as music present in the background. For streaming systems, to have minimal latency, bi-directional recurrent layers or temporal convolutional layers shall not be used in the model. For on-device systems, the model must be tiny and fast to require minimal budget in CPU and memory.

To achieve these goals, we implemented a new architecture which we call VoiceFilter-Lite[1]. In this architecture, the voice filtering model operates as a frame-by-frame frontend signal processor to enhance the features consumed by the speech recognizer, without reconstructing audio signals from the features. The key novel contributions of this work include: (1) A system to perform speech separation directly on ASR input features; (2) An asymmetric loss function to penalize over-suppression during training, to make the model harmless under various acoustic environments (Section 3.3); (3) An adaptive suppression strength mechanism to adapt to different noise conditions (Section 3.4). We also provided several approaches to make the model tiny and fast to meet strict on-device production requirements, such as limiting the model topology and quantizing the model to 8-bit integer format.

2. Review of the VoiceFilter system

Our new architecture shares many common designs with our previous VoiceFilter system [11], and here we give a brief introduction of the VoiceFilter base system.

At inference time, the VoiceFilter system takes two audio as input: noisy audio to be enhanced, and reference audio which is from the target speaker. A pre-trained speaker encoder LSTM [8] is used to produce the speaker-discriminative embedding, *a.k.a.* d-vector, from the reference audio. The time-frequency spectrogram of the noisy audio is computed via short-time Fourier transform (STFT), which first goes through convolutional layers, then is frame-wise concatenated with the d-vector, and finally goes through LSTM and fully connected layers to predict a time-frequency soft mask. The mask is element-wise multiplied to the spectral magnitude within the spectrogram to reconstruct enhanced audio via inverse STFT [13].

At training time, the noisy audio is generated by summing the waveforms of an interference audio and the clean audio, where the clean audio is from the same speaker as the reference audio, and the interference audio is from a different speaker. The VoiceFilter network is trained by minimizing a loss function which measures the difference between the clean spectrogram and the masked spectrogram.

[1]More details available at: https://google.github.io/speaker-id/publications/VoiceFilter-Lite

http://dx.doi.org/10.21437/Interspeech.2020-1193

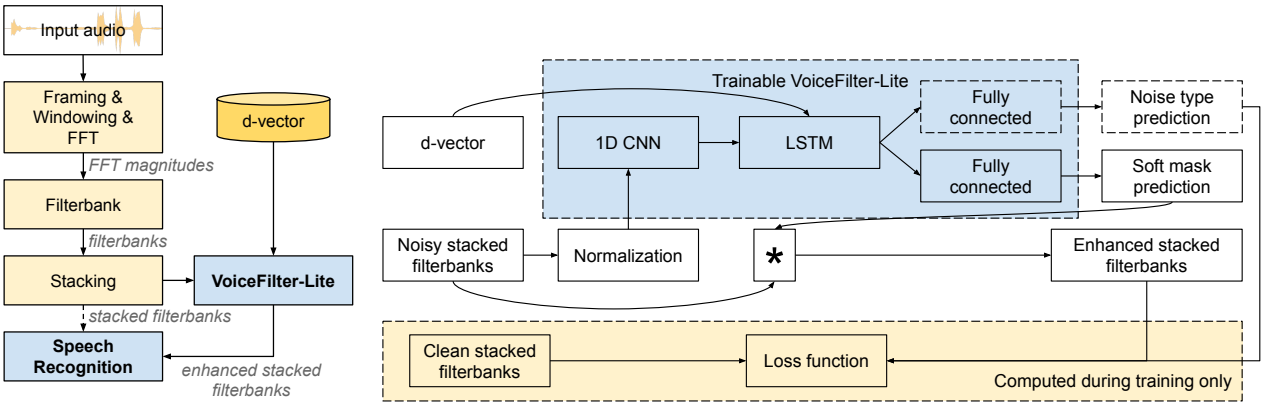

Figure 1: *VoiceFilter-Lite architecture, assuming using stacked filterbank energies as inputs and outputs. (a) Integration with ASR. The dashed arrow indicates the original connection without VoiceFilter-Lite. (b) Neural network topology of the VoiceFilter-Lite model.*

3. VoiceFilter-Lite

3.1. Integration with ASR

Our new VoiceFilter-Lite model focuses on improving ASR performance. Thus it is unnecessary to reconstruct any audio waveform from the masked spectrogram via inverse STFT. Instead, VoiceFilter-Lite operates as a frame-by-frame frontend signal processor that directly takes acoustic features as input, and outputs enhanced acoustic features. Here the acoustic features are exactly the same features being consumed by ASR models, such that the VoiceFilter-Lite model does not even need its own STFT operator. Since our ASR models take stacked log Mel-filterbank energies[2] as input, our VoiceFilter-Lite model has three integration options: (1) Takes FFT magnitudes as input, and outputs enhanced FFT magnitudes, which will be used to compute filterbank energies; (2) Takes filterbank energies as input, and outputs enhanced filterbank energies; (3) Directly takes stacked filterbank energies as input. For example, the last option is demonstrated in Fig. 1a. One benefit of this design is that the VoiceFilter-Lite model is independent from the ASR model, such that it can be easily bypassed when the d-vector is not available from the user, unlike systems such as [14].

3.2. Model topology

In Fig. 1b, we show the neural network topology of the VoiceFilter-Lite model assuming we use stacked filterbank energies as the inputs. Although it looks similar to the previous work described in [11], since VoiceFilter-Lite is designed for streaming ASR, we have made several changes to guarantee minimal latency: (1) We limit the convolutional layers to be 1D instead of 2D, meaning the convolutional kernels are for the frequency dimension only; (2) The LSTM layers must be uni-directional, and must be able to take streaming inputs. In practice, since frequency-only 1D-CNN is not as powerful as 2D-CNN, most of our models actually remove these CNN layers and purely consist of LSTM layers.

We assume the d-vector is available at runtime without additional computational cost. In a real application, users are usually prompted to follow an *enrollment* process [15, 16] before enabling speaker verification or voice filtering. During this one-off enrollment process, the d-vector is computed from the target user's recordings, and stored on the user's device.

At training time, the noisy audio is generated by mixing interference audio sources with the waveform of the clean audio, which is similar to [11]. However, in this work, to make our VoiceFilter-Lite model robust to various noise conditions, we add more variations to the noisification process: (1) the interference audio sources can be either speech from other speakers, or non-speech noise such as ambient noise or background music — we can also train the VoiceFilter-Lite model to predict the type of noise, which will be discussed in Section 3.4; (2) the noises can be applied through either additive or reverberant operations [17]; (3) the signal-to-noise ratio (SNR) is also random within 1dB to 10dB.

3.3. Asymmetric loss

Modern ASR models such as [12] are usually trained with extensively augmented data, such as multistyle training (MTR) [18, 19, 17] and SpecAugment [20]. Such ASR models already have great robustness against noise. If we add a voice filtering component to an existing ASR system, we must guarantee that the ASR performance does not degrade for any noise condition, which is a very challenging task. In fact, our early experiments show that when non-speech noise is present, the ASR performance could significantly degrade when we enable the voice filtering model. We observed that most of the degradation in Word Error Rate (WER) is from false deletions, which indicates significant **over-suppression** by the voice filtering model.

To overcome the over-suppression problem, we propose a new loss function for masking-based speech separation or enhancement, named **asymmetric loss**. Let $S_{\text{cln}}(t,f)$ and $S_{\text{enh}}(t,f)$ denote the clean and enhanced time-frequency spectrogram related features[3], respectively. A conventional L2 loss is defined as:

$$L = \sum_t \sum_f \Big(S_{\text{cln}}(t,f) - S_{\text{enh}}(t,f)\Big)^2. \quad (1)$$

We would like to be more tolerant of under-suppression errors, and less tolerant of over-suppression errors. Thus, we de-

[2]We may simply use "filterbanks" to refer to log Mel-filterbank energies in this paper. We use $\log(1+x)$ to avoid negative values, where 1 is a small value compared with the scale of the energies.

[3]Here the spectrogram can be either FFT magnitudes, filterbank energies, or stacked filterbank energies. The FFT magnitudes can also be power-law compressed like in [21].

fine an asymmetric penalty function g_{asym} with penalty factor $\alpha > 1$:

$$g_{\text{asym}}(x, \alpha) = \begin{cases} x & \text{if } x \leqslant 0; \\ \alpha \cdot x & \text{if } x > 0. \end{cases} \quad (2)$$

Then the asymmetric L2 loss function can be defined as:

$$L_{\text{asym}} = \sum_t \sum_f \Big(g_{\text{asym}}\big(S_{\text{cln}}(t,f) - S_{\text{enh}}(t,f), \alpha\big) \Big)^2. \quad (3)$$

3.4. Adaptive suppression strength

As mentioned before, modern ASR models are usually already robust against non-speech noise. Having a voice filtering model that performs an additional step of feature masking will often harm the ASR performance. Even with asymmetric loss, while our model significantly improves ASR performance under speech noise, it can still degrade ASR performance under non-speech noise.

One way to mitigate the performance degradation is to have an additional compensation to the over-suppression at inference time. Let $S_{\text{in}}^{(t)}$ and $S_{\text{enh}}^{(t)}$ denote the input and enhanced spectrogram related features at time t, respectively. The final compensated output would be:

$$S_{\text{out}}^{(t)} = w \cdot S_{\text{enh}}^{(t)} + (1 - w) \cdot S_{\text{in}}^{(t)}. \quad (4)$$

Here w is the suppression strength. When $w = 0$, voice filtering is completely disabled; and when $w = 1$, there is no compensation.

In practice, instead of using a fixed value of w, we wish to use a larger $w^{(t)}$ when the voice filtering model improves ASR, and a smaller $w^{(t)}$ when it hurts ASR. Thus, we add a second binary classification output to the model, which predicts whether a feature frame is from overlapped speech (class label 1) or not (class label 0) as shown in Fig. 1b. If we denote the noise type prediction as $f_{\text{adapt}}(S_{\text{in}}^{(t)}) \in [0, 1]$, the adaptive suppression strength at time t can be defined as

$$w^{(t)} = \beta \cdot w^{(t-1)} + (1 - \beta) \cdot \big(a \cdot f_{\text{adapt}}(S_{\text{in}}^{(t)}) + b\big), \quad (5)$$

where $a > 0$ and $b \geqslant 0$ define a linear transform, and $0 \leqslant \beta < 1$ is a moving average coefficient to make the suppression more smooth.

3.5. Model quantization

Raw TensorFlow graphs store network parameters in 32-bit floating point values, and are not well optimized for on-device inference. We quantize the network parameters to 8-bit integers with dynamic range quantization [22, 23], and serialize the models to the FlatBuffer TensorFlow Lite format. This helps us significantly reduce memory cost, and make better use of the optimized hardware instructions for integer arithmetic.

4. Experiments

4.1. Metrics

Our VoiceFilter-Lite model focuses on improving on-device ASR, thus we use the Word Error Rate (WER) as our metric. Specifically, we care about the WER before and after applying VoiceFilter-Lite under various noise conditions. Since the VoiceFilter-Lite model does not reconstruct any waveform from the enhanced acoustic features, and given the focus on speech recognition, we are not going to report metrics that have been widely used for conventional source separation, such as signal-to-noise ratio (SNR) or source-to-distortion ratio (SDR) [24, 25].

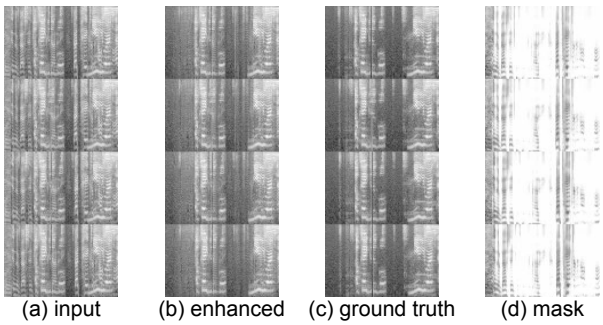

Figure 2: *Example stacked filterbanks from a segment of 3 seconds during training. x-axis is time and y-axis is filterbank.*

4.2. Model and data

In our experiments, the VoiceFilter-Lite model has 3 LSTM layers, each with 512 nodes, and a final fully connected layer with sigmoid activation. If noise type prediction is enabled, it has another two feedforward layers, each with 64 nodes, and is trained with a hinge loss [26] for binary classification. The training data consist of: (1) The LibriSpeech training set; and (2) a vendor-collected dataset of realistic speech queries. The d-vectors are generated with the same speaker encoder LSTM that has been used by [27] and [28].

For the WER evaluations, we use different testing sets for different ASR models, as described in Section 4.3 and Section 4.4, respectively. For all evaluations, we apply different noise sources and room configurations to the data. We use "Clean" to denote the original non-noisified data, although they could be quite noisy already. Our non-speech noise source consists of ambient noises recorded in cafes, cars, and quiet environments, as well as music or other types of sounds; the speech noise source is a distinct development set without overlapping speakers from the testing set. We evaluate with two types of room conditions: "Additive" means directly adding the noise waveform to the clean waveform; "Reverb" consists of 3 million convolutional room impulse responses (RIR) generated by a room simulator [17]. When applying the noises, the SNR is drawn from a uniform distribution from 1dB to 10dB for both additive and reverberant conditions.

4.3. Results on LibriSpeech

First, we use an RNN-transducer [29] based ASR model trained on the LibriSpeech training set [12], and compute the WER on LibriSpeech as well ("test-clean" and "test-other"). The results are reported in Table 1. Without voice filtering, the ASR model has a WER of 8.6% on the LibriSpeech testing set (5.2% on the "test-clean" subset). But after applying noises to the testing data, the WER becomes much worse.

With a VoiceFilter-Lite model, no matter which feature we use as input, the WER on the noisy conditions can be significantly reduced, but often hurting the WER in the clean condition. By training with asymmetric L2 loss, or applying a fixed suppression strength, the WER degradation in the clean case could be mitigated, at the cost of losing some performance gains in noisy conditions. For example, the filterbank-based model provides 47.3% (rel. 60.7%) WER improvements under additive speech noise conditions, without hurting WER in the clean case.

Table 1: *WER (%) for VoiceFilter-Lite models. ASR is trained and evaluated on LibriSpeech.*

Feature	Loss	Suppression strength	Clean	Non-speech noise		Speech noise		Size
				Additive	Reverb	Additive	Reverb	
No voice filtering			8.6	35.7	58.5	77.9	79.3	N/A
FFT magnitude	L2	$w = 1.0$	9.1	21.5	48.3	25.5	54.2	6.8 MB
	asym L2, $\alpha = 10$	$w = 1.0$	8.8	24.1	50.8	35.5	60.6	
Filterbank	L2	$w = 1.0$	9.3	23.4	48.9	25.4	55.6	5.8 MB
	asym L2, $\alpha = 10$	$w = 1.0$	8.6	24.8	49.8	30.6	58.4	
Stacked filterbank	L2	$w = 1.0$	8.9	22.2	48.2	23.5	53.7	6.8 MB
	asym L2, $\alpha = 10$	$w = 1.0$	8.8	23.9	49.7	30.6	57.8	
		$w = 0.6$	8.6	24.4	50.7	42.0	60.2	

Table 2: *WER (%) for VoiceFilter-Lite models. ASR is trained on multi-domain data, and evaluated on vendor-collected speech queries.*

Feature	Loss	Suppression strength	Clean	Non-speech noise		Speech noise		Size
				Additive	Reverb	Additive	Reverb	
No voice filtering			15.2	21.1	29.1	56.5	53.8	N/A
FFT magnitude	L2	$w = 1.0$	15.4	27.0	36.9	25.1	36.8	6.8 MB
	asym L2, $\alpha = 10$	$w = 1.0$	15.2	22.6	31.2	32.0	37.8	
Filterbank	L2	$w = 1.0$	15.3	28.5	38.3	26.5	38.5	5.8 MB
	asym L2, $\alpha = 10$	$w = 1.0$	15.3	26.6	35.6	27.5	37.4	
Stacked filterbank	L2	$w = 1.0$	16.7	26.8	36.2	26.8	37.4	6.8 MB
	asym L2, $\alpha = 10$	$w = 1.0$	15.8	25.7	34.4	27.4	36.7	
		$w = 0.3$	15.2	21.7	29.6	42.1	43.6	
		Adaptive $w^{(t)}$	15.3	21.3	29.3	28.8	37.2	
			15.4	21.1	29.0	31.4	39.1	2.2 MB

4.4. Results on realistic speech queries

Although the above results demonstrated the success of VoiceFilter-Lite on LibriSpeech, we would also like to evaluate on more realistic data. We trained another on-device streaming RNN-transducer ASR model [12] using data from multiple domains including YouTube and anonymized voice search, then evaluate the WER on a vendor-collected dataset of realistic speech queries. This dataset was collected via a Web-based tool, and consists of about 20 thousand utterances from 230 speakers. The task here is much more challenging than LibriSpeech because: (1) There are more variations of prosody, sentiment, accent, and acoustic conditions in real speech queries than read speech; (2) We are evaluating across the domain, since the type of evaluation data is not represented in ASR training.

From the results shown in Table 2, we can see that, if the VoiceFilter-Lite model is trained with L2 loss, although the WERs on speech noise are greatly reduced, the WERs on non-speech noise largely degrade. This observation is consistent no matter what features we use for VoiceFilter-Lite. A model like this could not be used in production, since it harms performance in certain conditions. However, with the asymmetric L2 loss introduced in Section 3.3, the degradation on non-speech noise is much smaller, although we also see less improvement on speech noise at the same time.

For models using stacked filterbanks as features, we also experimented with suppression compensation. If we use a fixed strength $w = 0.3$, the degradation on non-speech noise is further reduced, but the improvement on speech noise is also compromised. If we use adaptive suppression strength with a moving average coefficient $\beta = 0.8$ (2nd last row in Table 2), we observe almost no WER degradation on clean and non-speech noise conditions, and still a large improvement on speech noise conditions — 27.7% (rel. 49.0%) WER improvement under additive speech noise condition. This meets our goal of having an **always harmless and sometimes helpful** model that is safe to use in real applications. By reducing each LSTM layer to only 256 nodes, the model size becomes 2.2 MB (last row in Table 2), but still with similar WER performance.

4.5. Discussions

For our three different features, *i.e.* the 513-dim FFT magnitudes, the 128-dim log Mel-filterbank energies, and the 512-dim stacked-by-4 log Mel-filterbank energies, our results are quite similar. However, we argue that using stacked filterbanks as model inputs and outputs is largely preferred: (1) First, the stacked filterbanks contain more contextual information than single-frame FFT or filtebanks; (2) Second, since frame subsampling is usually applied to reduce redundant information from frame stacking, the network runs less often when taking stacked filterbanks as input, which largely reduces CPU cost with the same number of network parameters. Examples of stacked filterbanks during training are visualized in Fig. 2.

5. Conclusions

In this paper, we described VoiceFilter-Lite, a tiny and fast model that performs targeted voice separation in a streaming fashion, as part of an on-device ASR system. Since modern ASR models are already trained with a diverse range of noise conditions, we need to guarantee that the voice separation model does not hurt the ASR performance, especially under non-speech noise, and reverberant room conditions. We achieved this by training our model with an asymmetric loss function, and applying an adaptive suppression strength at runtime. Combining these novel efforts, we developed a 2.2 MB model that has no WER degradation on the clean and non-speech noise conditions we measured, while largely improving ASR performance on overlapped speech.

6. Acknowledgements

The authors would like to thank Philip Chao, Sinan Akay, John Han, Stephen Wu, Yiteng Huang, Jaclyn Konzelmann and Nino Tasca for the support and helpful discussions.

7. References

[1] E. C. Cherry, "Some experiments on the recognition of speech, with one and with two ears," *The Journal of the acoustical society of America*, vol. 25, no. 5, pp. 975–979, 1953.

[2] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 31–35.

[3] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 246–250.

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

[5] M. Kolbæk, D. Yu, Z.-H. Tan, and 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 (TASLP)*, vol. 25, no. 10, pp. 1901–1913, 2017.

[6] K. Zmolikova, M. Delcroix, K. Kinoshita, T. Higuchi, A. Ogawa, and T. Nakatani, "Speaker-aware neural network based beamformer for speaker extraction in speech mixtures," in *Proc. Interspeech*, 2017.

[7] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[8] L. Wan, Q. Wang, A. Papir, and I. Lopez Moreno, "Generalized end-to-end loss for speaker verification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4879–4883.

[9] J. Wang, J. Chen, D. Su, L. Chen, M. Yu, Y. Qian, and D. Yu, "Deep extractor network for target speaker recovery from single channel speech mixtures," *arXiv preprint arXiv:1807.08974*, 2018.

[10] M. Delcroix, K. Zmolikova, K. Kinoshita, A. Ogawa, and T. Nakatani, "Single channel target speaker extraction and recognition with speaker beam," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5554–5558.

[11] Q. Wang, H. Muckenhirn, K. Wilson, P. Sridhar, Z. Wu, J. R. Hershey, R. A. Saurous, R. J. Weiss, Y. Jia, and I. Lopez Moreno, "VoiceFilter: Targeted voice separation by speaker-conditioned spectrogram masking," in *Proc. Interspeech*, 2019, pp. 2728–2732.

[12] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[13] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[14] P. Denisov and N. T. Vu, "End-to-end multi-speaker speech recognition using speaker embeddings and transfer learning," *arXiv preprint arXiv:1908.04737*, 2019.

[15] N. Jensen, "More ways to fine tune Google Assistant for you," https://www.blog.google/products/assistant/more-ways-fine-tune-google-assistant-you/, April 2020.

[16] Q. Wang and I. Lopez Moreno, "Version control of speaker recognition systems," *arXiv preprint arXiv:2007.12069*, 2020.

[17] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," in *Proc. Interspeech*, 2017, pp. 379–383.

[18] R. Lippmann, E. Martin, and D. Paul, "Multi-style training for robust isolated-word speech recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 12. IEEE, 1987, pp. 705–708.

[19] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

[20] 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.

[21] 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 *International Workshop on Acoustic Signal Enhancement (iWAENC)*. IEEE, 2018, pp. 366–370.

[22] R. Alvarez, R. Prabhavalkar, and A. Bakhtin, "On the efficient representation and execution of deep acoustic models," *arXiv preprint arXiv:1607.04683*, 2016.

[23] Y. Shangguan, J. Li, Q. Liang, R. Alvarez, and I. McGraw, "Optimizing speech recognition for the edge," *arXiv preprint arXiv:1909.12408*, 2019.

[24] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

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

[26] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[27] Y. Jia, R. J. Weiss, F. Biadsy, W. Macherey, M. Johnson, Z. Chen, and Y. Wu, "Direct speech-to-speech translation with a sequence-to-sequence model," in *Proc. Interspeech*, 2019, pp. 1123–1127.

[28] S. Ding, Q. Wang, S.-Y. Chang, L. Wan, and I. Lopez Moreno, "Personal VAD: Speaker-conditioned voice activity detection," in *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*, 2020, pp. 433–439.

[29] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.