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

Lightweight Online Noise Reduction on Embedded Devices using Hierarchical Recurrent Neural Networks

H. Schröter[1], *T. Rosenkranz*[2], *A. N. Escalante-B*[2], *P. Zobel*[1], *A. Maier*[1]

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Pattern Recognition Lab
[2] Sivantos GmbH, Research and Development, Erlangen, Germany

hendrik.m.schroeter@fau.de

Abstract

Deep-learning based noise reduction algorithms have proven their success especially for non-stationary noises, which makes it desirable to also use them for embedded devices like hearing aids (HAs). This, however, is currently not possible with state-of-the-art methods. They either require a lot of parameters and computational power and thus are only feasible using modern CPUs. Or they are not suitable for online processing, which requires constraints like low-latency by the filter bank and the algorithm itself.

In this work, we propose a mask-based noise reduction approach. Using hierarchical recurrent neural networks, we are able to drastically reduce the number of neurons per layer while including temporal context via hierarchical connections. This allows us to optimize our model towards a minimum number of parameters and floating-point operations (FLOPs), while preserving noise reduction quality compared to previous work. Our smallest network contains only $5\,\mathrm{k}$ parameters, which makes this algorithm applicable on embedded devices. We evaluate our model on a mixture of EUROM and a real-world noise database and report objective metrics on unseen noise.

Index Terms: speech enhancement, noise reduction, recurrent neural networks, embedded devices

1. Introduction

Noise reduction (NR) aims at reducing unwanted environmental noise, like street noise, and enhances a superimposed speech signal. NR is an important feature of modern hearing aids or hearing assistance devices. Recent contributions to deep-learning based monaural speech enhancement [1, 2, 3, 4, 5] result in a huge improvement over conventional noise suppression approaches [6, 7]. This makes it desirable to incorporate these approaches into HAs. However, these algorithms employing deep neural networks have great demand on both memory and computational power. Furthermore, many algorithms process the noisy signal in an offline fashion [8, 2, 9, 10, 11] or introduce large delays, which is not viable on HAs. According to Jeremy et al. [12], the maximum delay of what is typically acceptable is $10\,\mathrm{ms}$. Having an open acoustic coupling, a greater delay introduces annoying comb filter effects due to the superposition of the processed and direct signal.

The approaches that are close to our real-time and online processing constraints were proposed by Valin [13] and Aubreville [3]. Valin et al. [13] uses an RNN processing $20\,\mathrm{ms}$ windows with a $50\,\%$ overlap operating at a sampling rate of $48\,\mathrm{kHz}$. To reduce the model complexity they used a bark like scaling, which further lowers the number of input and output units. This resulted in a network containing $88.5\,\mathrm{k}$ parameters and about $40\,\mathrm{MFLOPs}$ per second. While this algorithm is real-time capable on a Raspberry Pi and processes the data in an online fashion, the introduced delay is greater than $20\,\mathrm{ms}$ which is too long for our requirements.

Aubreville et al. [3] employed a hearing instrument-grade filter bank that introduced a combined latency of analysis and synthesis of about $6\,\mathrm{ms}$. Additionally, they included future context of $2\,\mathrm{ms}$ resulting in an overall latency of $8\,\mathrm{ms}$. However, they predicted Wiener gains using a fully connected network containing about $28.6\,\mathrm{M}$ parameters, resulting in about $57.3\,\mathrm{GFLOPs}$ per second for the algorithm only, not including filter bank computations.

In this work, we take low latency requirements ($\leq 10\,\mathrm{ms}$) into account and furthermore focus on a parameter and FLOPs reduction. To achieve our goals, we employ a uniform polyphase filter bank with a low spectral resolution. We process our data in an about $6\,\mathrm{ms}$ frame basis with a $1\,\mathrm{ms}$ hop (Sec. 2). While RNN cells, like gated recurrent units (GRUs) or long short-term memory (LSTM) cells, are able to capture long and short term dependencies, they require a sufficient amount of parameters and are hard to train. To be able to reduce the number of parameters and thus hidden state of the recurrent state, we use a hierarchical structure to incorporate a short-term temporal context of $\pm 1\,\mathrm{ms}$. This allows us to employ GRU cells with down to only 12 hidden units. We report results on the EUROM database using 260 German sentences and 49 real-world noise signals recorded with hearing aid equipment in Sec. 3. Furthermore, we provide a comparison with conventional approaches as well as previous work that employs the same processing toolchain. We analyse the complexity of our models in Sec. 4 and provide calculation basis and assumptions for the FLOP estimation.

2. Signal Processing Toolchain

We use a standard uniform polyphase filter bank to transform the time domain signal into time/frequency (TF) domain. Operating on $24\,\mathrm{kHz}$ sampling rate, the analysis window processes the input signal in an about $6\,\mathrm{ms}$ frame basis with an offset (hop) of $1\,\mathrm{ms}$. This filter bank ensures our low-latency requirements, but results in a low resolution spectral representation with 48 frequency bins.

The signal block diagram of the noise reduction system is shown in Fig. 1. We transform the complex filter bank representation into decibel scale and normalize it using exponential averaging. The noise reduction itself is performed on a bark compressed spectral representation via a real valued mask. The RNN is trained using the magnitude spectral approximation (MSA) loss [14].

http://dx.doi.org/10.21437/Interspeech.2020-1131

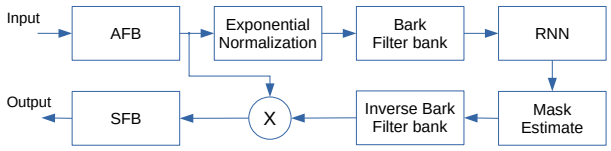

Figure 1: *Processing toolchain. AFB/SFB depict the analysis and synthesis filter banks.*

2.1. Normalization

Normalization is a crucial part of neural network training, as it tremendously helps with convergence and generalization and reduces the initialization impact. Furthermore, since the perceived loudness by a human is scaled logarithmically, we first transform the complex filter bank representation into decibel scale and clip at $-100\,\mathrm{dB}$.

$$X_{\mathrm{dB}}[t,f] = 10 \cdot \log_{10}(\max(|X[t,f]|^2, 10^{-10}))\,, \quad (1)$$

where t represents a time step in TF domain and f a frequency band. We then normalize the spectrogram per frequency bin to zero mean and unit variance:

$$X_{\mathrm{norm}}[t,f] = \frac{X_{\mathrm{dB}}[t,f] - \hat{\mu}[t,f]}{\sqrt{\hat{\sigma}^2[t,f]}}\,, \quad (2)$$

where $\hat{\mu}$ and $\hat{\sigma}$ are the estimated mean and variance. That is, we can calculate mean estimates $\hat{\mu}$ and sample square estimates $\hat{s}$ via an exponential decay [15]:

$$\hat{\mu}[t,f] = \alpha\hat{\mu}[t-1,f] + (1-\alpha)X_{\mathrm{dB}}[t,f]\,, \quad (3)$$

$$\hat{s}^2[t,f] = \alpha\hat{s}^2[t-1,f] + (1-\alpha)(X_{\mathrm{dB}}[t,f])^2\,, \quad (4)$$

and get the variance estimate

$$\hat{\sigma}^2[t,f] = \hat{s}^2[t,f] - \hat{\mu}^2[t,f]\,. \quad (5)$$

Viiki et al. [15] suggested a normalization period τ of approximately $1\,\mathrm{s}$. With our sampling rate in the filter bank domain, this corresponds to $\alpha = \exp(-\Delta t/\tau) \approx 0.999$. We furthermore evaluated our model with and without variance normalization in experiment 3.1. I.e., we just assumed that the input had unit variance and skipped the square and square-root computations. Xia et al. [16] also used exponentially decaying normalization and compared it with global mean/variance normalization based on the training set. They recommend the online approach based on exponential decay and a normalization period of $3\,\mathrm{s}$.

2.2. Bark scale

Instead of performing the mask based noise reduction directly on the uniform filter bank representation, we further reduce the input and output dimensions using a bark like scaling of the frequency bands. Thus, we can take advantage of the fact that human frequency perception is also on a logarithmic scale and use a coarser frequency resolution for the higher frequencies. That is, we reduce the normalized spectrogram from 48 channels to 16 bands using rectangular bands, ensuring that the first 8 bark bands until $2\,\mathrm{kHz}$ only have 1 frequency channel and follow the bark scale for higher frequencies. Furthermore, the network only produces a mask with 16 bins that is transformed into linear frequency scale via an inverse operation. This allows to reduce the network size tremendously and we found that it helps a lot with convergence for very small networks.

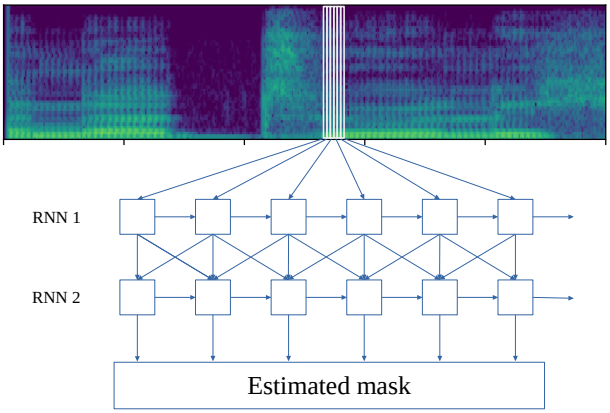

Figure 2: *Hierarchical RNN architecture. The second layer RNN includes temporal context of the previous, current and next time step.*

2.3. Hierarchical RNN

Temporal context is essential for noise reduction algorithms. Aubreville et al. [3] showed that a temporal look-back context $\gg 30\mathrm{ms}$ is required to differentiate a fricative with little energy at low frequencies from noise only. They furthermore showed that additionally to past context, future context also results in a better noise reduction. However, future context always introduces a non-desirable delay, which is why they limited their future context to $2\,\mathrm{ms}$.

Past context can obviously be incorporated using the hidden state of the RNN. Nonetheless, the size of the hidden state, i.e., the number of neurons, limits the amount of context that can be stored. Therefore, to allow the network to focus on the long-term temporal context such as being in a word or phoneme, we incorporate a temporal context of $\pm 1\,\mathrm{ms}$, denoted as hierarchical context (HC). But instead of including the short-term context as input for the first RNN layer (C-RNN), we provide a context window of the first layer output as input for the second layer (HC-RNN) as shown in Fig. 2. This hierarchical short-term context inclusion is similar to fully convolutional networks that have an increasing receptive field for deeper layers due to stride > 1. We show in our experiments that the hierarchical structure using HC outperforms a standard RNN with early fusion.

2.4. Network Training

Several loss functions have been proposed in recent years. In this work, we only focus on real-valued mask-based losses for performance and robustness reasons. These loss functions can be broadly divided into two categories: Mask approximation (MA) and signal approximation. For the former, common mask targets are ideal ratio mask (IRM), ideal amplitude mask (IAM) or a "Wiener filter like" mask (WF) [1]. All of these have slightly different properties, while WF is optimal w.r.t. the maximum SNR, if speech and noise are uncorrelated. Aubreville et al. successfully used a WF like mask in a hearing aids setting.

The MA loss is defined as

$$L_{\text{MA}} = \sum_{t,f} (|\,|M[t,f]| - \hat{M}[t,f]\,|^2)\,, \tag{6}$$

where $\hat{M}$ is the mask estimate and M the target mask, e.g. WF. Weninger et al. [14] didn't compute the loss based on the mask, but rather forced the network to output a mask that was directly applied to the noisy signal. The loss then was computed based on the magnitude of the resulting clean and enhanced spectrograms. This loss is called magnitude spectrum approximation (MSA) and defined as

$$L_{\text{MSA}} = \sum_{t,f} (|\,|S[t,f]| - |X[t,f]| \odot \hat{M}[t,f]\,|^2)\,, \tag{7}$$

where $\odot$ is a point-wise multiplication. We noticed that especially for noisy conditions, MSA loss outperforms the MA loss like WF. Additionally, we provide a comparison with a combined MA-MSA loss in experiment 3.3. We trained the network with gated recurrent units (GRU) layers using sequences of at least 5 s, a batch size of 20 and Adam optimizer with a learning rate of 0.001 using the deep learning framework PyTorch [17].

3. Experiments

In this section, we present various experiments and evaluate them using the SI-SDR metric [18] and the difference of noisy to enhanced short-time objective intelligibility (STOI) [19] denoted as ΔSTOI. All networks (except from experiment 3.1) use a 2 layer GRU followed by a fully connected layer with sigmoid activation. We split our noise and speech corpus on original signal level in a train, validation (dev) and test set. We use the same splittings as [3, 5]. All results are based on the test set unless otherwise stated.

3.1. Experiment: Variance Normalization

While many studies use zero mean and unit variance normalization [3, 16] for the input, we found that normalizing to unit variance does not provide additional benefit. Tab. 1 shows STOI improvement of a simple 3 layer LSTM network with 48 hidden units each. There is no significant difference, which leads us to the assumption that the first network layer is able to model the input variance. Furthermore, the network only produces an output mask that is applied to the noisy spectrogram, the mask should be independent of the noisy spectrogram scale. Thus, for further experiments to minimize the computational effort, we only use zero mean normalization.

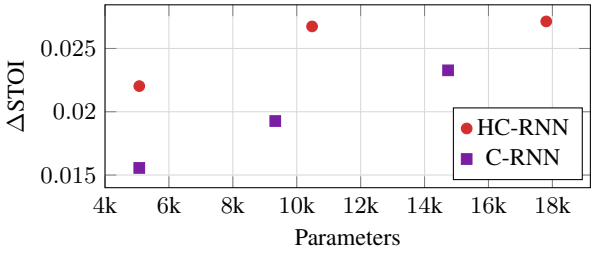

Figure 3: *Mean ΔSTOI scores for networks of different sizes.*

Table 1: *ΔSTOI dev set results when providing input with zero mean and unit variance, and zero mean only while keeping the original variance.*

Normalization	ΔSTOI
Zero mean & unit variance	0.0353
Zero mean	0.0347

3.2. Experiment: Complexity of short-term context via Hierarchical RNNs

In this experiment, we show that networks with hierarchical context structure (HC-RNN) are favorable w.r.t. ΔSTOI score. While HC-RNNs only need a small input layer, the second layer incorporates the short-term context of ± 1 frames and thus is larger. We compare these with networks that incorporate the short-term context in the first layer and thus, have a larger 1st layer than 2nd layer (C-RNN). Fig. 3 shows ΔSTOI scores of various network sizes. All network configurations are shown in table 2.
We can see that hierarchical RNN outperforms the conventional RNN that includes context in the first layer on a per parameter basis. This leads us to conclude that at least for these small numbers of parameters, it is beneficial to integrate the short-term context at later layer and thus shift the majority of parameters to deeper layers.

3.3. Experiment: Mask Approximation vs. Signal Approximation

Prior work reported that an MSA objective outperforms MA approaches like WF [1, 14]. However, we found that this heavily depends on the input SNR. The box plot in Fig. 4 shows that MSA outperforms MA with a WF like mask in noisy conditions (SNR $\leq$ 0). For SNRs $\geq$ 5, MA and MA-MSA loss results in better ΔSTOI scores. Since MA-MSA and MSA seem to perform similarly, we evaluated both for our smallest model with 5 k parameters in experiment 3.4.

3.4. Experiment: Objective evaluation and comparison with prior work

We compare our smallest HC-RNN model with various state-of-the-art results. As a baseline we chose recursive minimum tracking [20] and also provide results using FC-WG [3] and RNNoise [13] based on our test set. As one can see in Tab. 3,

Table 2: *Network configurations with input dimension **I**, hidden dimension **H** and number of parameters **P** for the different networks in figure 3. For all networks, the output layer is a fully connected layer with hidden dimension 16.*

	GRU Layer 1		GRU Layer 2		Total
Context	# **I**	# **H**	# **I**	# **H**	# **P**
C-RNN	48	16	16	16	5 072
C-RNN	48	24	24	24	9 382
C-RNN	48	32	32	32	14 736
HC-RNN	16	16	48	16	5 072
HC-RNN	16	24	72	24	10 480
HC-RNN	16	32	96	32	17 808

Table 3: *Metric results on the test set using our smallest HC-RNN model, RNNoise [13] and FC-WF [3]. Number of network parameters in brackets. SI-SDR and ΔSTOI are provided depending on the input SNR, the root mean squared error of the time-domain signal (RMSE) is averaged over all input SNRs. The recursive minimum tracking baseline [20] is also evaluated using an attenuation limit of* 14 dB.

Metric per SNR [dB]	SI-SDR [dB]					ΔSTOI					RMSE
	−5	0	5	10	20	−5	0	5	10	20	-
Model											
HC-RNN$_{MSA}$ (5 k)	**0.82**	5.46	9.71	**13.81**	**21.71**	**0.025**	0.038	0.027	0.015	0.003	0.022
HC-RNN$_{MA-MSA}$ (5 k)	0.56	5.18	9.30	13.43	21.68	0.023	**0.042**	0.036	0.024	**0.008**	0.022
RNNoise (87 k) [13]	−0.84	**6.46**	**10.14**	12.71	15.45	−0.005	0.032	0.032	0.024	0.007	**0.021**
FC-WF (25 M) [3]	−1.65	3.59	8.44	12.84	21.41	0.019	**0.042**	**0.039**	**0.027**	0.007	0.025
Baseline	−5.68	0.91	5.97	10.17	16.94	−0.144	−0.095	−0.059	−0.040	−0.026	0.028
Baseline$_{14\,\mathrm{dB}}$	−4.07	1.74	6.57	10.67	17.48	−0.006	−0.011	−0.015	−0.016	−0.015	0.027

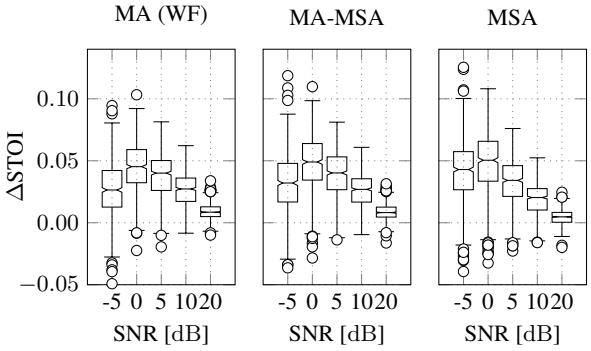

Figure 4: *ΔSTOI results using a hierarchical RNN with 25 k parameters trained with MSA loss, MA loss using a WF like mask and combination.*

the HC-RNN models performs similarly to prior studies, while the performance slightly drops for larger SNRs for the MSA loss. In any case, it still outperforms conventional methods like recursive minimum tracking [20], that is only able to provide an SDR improvement for SNRs in range 0 to 10.

4. Complexity analysis

The number of FLOPs per second of a GRU is given by

$$T \cdot 6N(M + N + 1) \tag{8}$$

where T are the number of time steps, M the input size and N the hidden size. In our case, we have $T = 1000$ steps per second, the smallest network has 16 hidden units in each layer, while the input size is 16 and 48 respectively. We count a separate operation for multiply and add. For activation functions like tanh or sigmoid, we assume a lookup table consuming 1 FLOP. Additionally with the fully connected output layer, this results in 10.0 MFLOPS. The normalization requires additional 0.14 MFLOPS, the bark scaling 48 kFLOPS. The filter bank is not considered here, since it is required also for other tasks and implemented via hardware. Compared to RNNoise which requires about 40 MFLOPS, we still have a considerable FLOPS reduction of a factor 4, while lowering the delay from ≥20 ms to 8 ms. Our model is able to run in real-time on a Raspberry Pi 3 using the non optimized Python front-end of the deep learning framework PyTorch.

5. Conclusion

We presented a real-time noise reduction approach for low-delay and low-computation requirements. Our overall delay is about 8 ms, consisting of about 6 ms filter bank analysis and synthesis, 1 ms future context and 1 ms processing. While we are able to obtain similar results to other real-time approaches like RNNoise, we reduce the number of parameters significantly to only 5 k and number of FLOPS to about 10 M.

6. References

[1] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 708–712.

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

[3] M. Aubreville, K. Ehrensperger, A. Maier, T. Rosenkranz, B. Graf, and H. Puder, "Deep denoising for hearing aid applications," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 361–365.

[4] J. Le Roux, G. Wichern, S. Watanabe, A. Sarroff, and J. R. Hershey, "Phasebook and friends: Leveraging discrete representations for source separation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 370–382, 2019.

[5] H. Schröter, T. Rosenkranz, A. N. E. B., M. Aubreville, and A. Maier, "CLCNet: Deep learning-based Noise Reduction for Hearing Aids using Complex Linear Coding," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[6] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[7] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on speech and audio processing*, vol. 9, no. 5, pp. 504–512, 2001.

[8] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder." in *Interspeech*, 2013, pp. 436–440.

[9] D. S. Williamson, "Monaural speech separation using a phase-aware deep denoising auto encoder," in *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2018, pp. 1–6.

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

[11] H. Zhao, S. Zarar, I. Tashev, and C.-H. Lee, "Convolutional-recurrent neural networks for speech enhancement," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2401–2405.

[12] J. Agnew and J. M. Thornton, "Just noticeable and objectionable group delays in digital hearing aids," *Journal of the American Academy of Audiology*, vol. 11, no. 6, pp. 330–336, 2000.

[13] J.-M. Valin, "A hybrid DSP/deep learning approach to real-time full-band speech enhancement," in *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2018, pp. 1–5.

[14] F. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," in *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2014, pp. 577–581.

[15] O. Viikki, D. Bye, and K. Laurila, "A recursive feature vector normalization approach for robust speech recognition in noise," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, vol. 2. IEEE, 1998, pp. 733–736.

[16] Y. Xia, S. Braun, C. K. A. Reddy, H. Dubey, R. Cutler, and I. Tashev, "Weighted speech distortion losses for neural-network-based real-time speech enhancement," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 871–875.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

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

[19] 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.

[20] E. Hänsler and G. Schmidt, *Acoustic echo and noise control: a practical approach*. John Wiley & Sons, 2005, vol. 40.