A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement

Jean-Marc Valin

Mozilla Corporation

Mountain View, CA, USA

jmvalin@jmvalin.ca

Abstract—**Despite noise suppression being a mature area in signal processing, it remains highly dependent on fine tuning of estimator algorithms and parameters. In this paper, we demonstrate a hybrid DSP/deep learning approach to noise suppression. A deep neural network is used to estimate ideal critical band gains, while a more traditional pitch filter attenuates noise between pitch harmonics. The approach achieves significantly higher quality than a traditional minimum mean squared error spectral estimator, while keeping the complexity low enough for real-time operation at 48 kHz on a low-power processor.**

I. INTRODUCTION

Noise suppression has been a topic of interest since at least the 70s. Despite significant improvements in quality, the high-level structure has remained mostly the same. Some form of spectral estimation technique relies on a noise spectral estimator, itself driven by a voice activity detector (VAD) or similar algorithm, as shown in Fig. 1. Each of the 3 components requires accurate estimators and are difficult to tune. For example, the crude initial noise estimators and the spectral estimators based on spectral subtraction [1] have been replaced by more accurate noise estimators [2], [3] and spectral amplitude estimators [4]. Despite the improvements, these estimators have remained difficult to design and have required significant manual tuning effort. That is why recent advances in deep learning techniques are appealing for noise suppression.

Deep learning techniques are already being used for noise suppression [5], [6], [7], [8]. Most of the proposed approaches target automatic speech recognition (ASR) applications, where latency and computational resources are not important factors. In the proposed approach we instead focus on real-time applications (e.g. video-conference) with low complexity. We also focus on full-band (48 kHz) speech. To achieve these goals we choose a hybrid approach (Sec. II), where we rely on proven signal processing techniques and use deep learning (Sec. III) to replace the estimators that have traditionally been hard to correctly tune. The approach contrasts with so-called *end-to-end* systems where most or all of the signal processing operations are replaced by machine learning. These end-to-end system have clearly demonstrated the capabilities of deep learning, but they often come at the cost of significantly increased complexity.

We show that the proposed approach has an acceptable complexity (Sec. IV) and that it provides better quality than more conventional approaches (Sec. V). We conclude in Sec. VI with directions for further improvements to this approach.
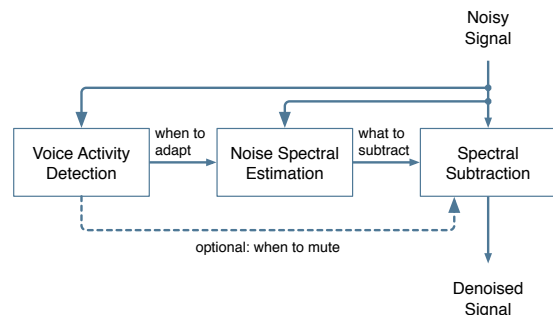

Figure 1. High-level structure of most noise suppression algorithms.

II. SIGNAL MODEL

We propose a hybrid approach to noise suppression. The goal is to use deep learning for the aspects of noise suppression that require careful tuning while using basic signal processing building blocks for parts that do not.

The main processing loop is based on 20 ms windows with 50% overlap (10 ms offset). Both analysis and synthesis use the same Vorbis window [9], which satisfies the Princen-Bradley criterion [10]. The window is defined as

$$w(n) = \sin\left[\frac{\pi}{2}\sin^2\left(\frac{\pi n}{N}\right)\right] , \tag{1}$$

where N is the window length.

The signal-level block diagram for the system is shown in Fig. 2. The bulk of the suppression is performed on a low-resolution spectral envelope using gains computed from a recurrent neural network (RNN). Those gains are simply the square root of the ideal ratio mask (IRM). A finer suppression step attenuates the noise between pitch harmonics using a pitch comb filter.

A. Band structure

In the approach taken by [5], a neural network is used to directly estimate magnitudes of frequency bins and requires a total of 6144 hidden units and close to 10 million weights to process 8 kHz speech. Scaling to 48 kHz speech using 20-ms frames would require a network with 400 outputs (0 to 20 kHz), which clearly results in a higher complexity than we can afford.
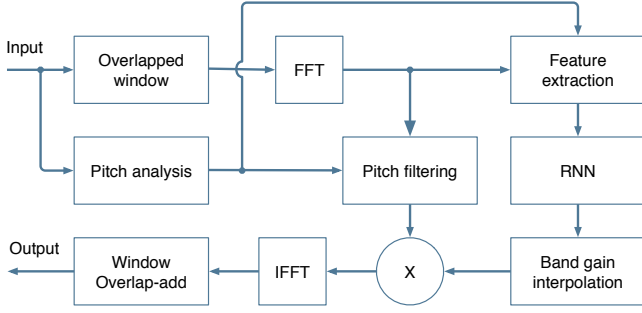
Figure 2. Block diagram.

One way to avoid the problem is to assume that the spectral envelopes of the speech and noise are sufficiently flat to use a coarser resolution than frequency bins. Also, rather than directly estimate spectral magnitudes, we instead estimate ideal critical band gains, which have the significant advantage of being bounded between 0 and 1. We choose to divide the spectrum into the same approximation of the Bark scale [11] as the Opus codec [12] uses. That is, the bands follow the Bark scale at high frequencies, but are always at least 4 bins at low frequencies. Rather than rectangular bands, we use triangular bands, with the peak response being at the boundary between bands. This results in a total of 22 bands. Our network therefore requires only 22 output values in the $[0, 1]$ range.

Let $w_b(k)$ be the amplitude of band b at frequency k, we have $\sum_b w_b(k) = 1$. For a transformed signal $X(k)$, the energy in a band is given by

$$E(b) = \sum_k w_b(k) |X(k)|^2 \ .$$ (2)

The per-band gain is defined as g_b

$$g_b = \sqrt{\frac{E_s(b)}{E_x(b)}} \ ,$$ (3)

where $E_s(b)$ is the energy of the clean (ground truth) speech and $E_x(b)$ is the energy of the input (noisy) speech. Considering an ideal band gain $\hat{g}_b$, the following interpolated gain is applied to each frequency bin k:

$$r(k) = \sum_b w_b(k) \hat{g}_b \ .$$ (4)

B. Pitch filtering

The main disadvantage of using Bark-derived bands to compute the gain is that we cannot model finer details in the spectrum. In practice, this prevents noise suppression between pitch harmonics. As an alternative, we can use a comb filter at the pitch period to cancel the inter-harmonic noise in a similar way that speech codec post-filters operate [13]. Since the periodicity of speech signal depends heavily on frequency (especially for 48 kHz sampling rate), the pitch filter operates in the frequency domain based on a per-band filtering coefficient α_b. Let $P(k)$ be the DFT of the signal $x(n - T)$ delayed by the pitch period T, the filtering is performed

by computing $X(k) + \alpha_b P(k)$ and then renormalizing the resulting signal to have the same energy in each band as the original signal $X(k)$.

The pitch correlation for band b is defined as

$$p_b = \frac{\sum_k w_b(k) \Re[X(k) P^*(k)]}{\sqrt{\sum_k w_b(k) |X(k)|^2 \cdot \sum_k w_b(k) |P(k)|^2}} \ ,$$ (5)

where $\Re[\cdot]$ denotes the real part of a complex value and $\cdot^*$ denotes the complex conjugate. Note that for a single band, then (5) is equal to the pitch correlation.

It is hard to derive the optimal values for the filtering coefficient and we values that minimizes mean squared error does not appear to be perceptually optimal. Instead, we use a heuristic based on the following constraints and observations. Since noise causes a decrease in the pitch correlation, we do not expect p_b to be greater than g_b *on average*, so for any band that has $p_b \geq g_b$, we want to use $\alpha_b=1$. When there is no noise, we do not want to distort the signal, so when $g_b = 1$, we use $\alpha_b = 0$. Similarly, when $p_b = 0$, we have no pitch to enhance, so $\alpha_b = 0$. Using the following expression for the filtering coefficient respects all these constraints with smooth behavior between:

$$\alpha_b = \min\left[1, \ \frac{p_b^2 (1 - g_b^2)}{(1 - p_b^2) g_b^2}\right]$$ (6)

C. Feature extraction

It only makes sense for the input of the network to include the log spectrum of the noisy signal based on the same bands used for the output. To improve the conditioning of the training data, we apply a DCT on the log spectrum, which results in 22 Bark-frequency cepstral coefficients (BFCC). In addition to these, we also include the temporal derivative and the second temporal derivative of the first 6 BFCCs. Since we already need to compute the pitch in (5), we compute the DCT of the pitch correlation across frequency bands and include the first 6 coefficients in our set of features. At last, we include the pitch period as well as a spectral non-stationarity metric that can help in speech detection. In total we use 42 input features to the RNN.

Unlike the features typically used in speech recognition, these features do not use cepstral mean normalization and do include the first cepstral coefficient. The choice is deliberate given that we have to track the absolute level of the noise, but it does make the features sensitive to the absolute amplitude of the signal and to the channel frequency response. This is addressed in Sec. III-A.

III. Deep Learning Architecture

The neural network closely follows the traditional structure of noise suppression algorithms, as shown in Fig. 3. The design is based on the assumption that the three recurrent layers are each responsible for one of the basic components from Fig. 1. Of course, in practice the neural network is free to deviate from this assumption (and likely does to some extent). It includes a total of 215 units, 4 hidden layers, with the largest layer having
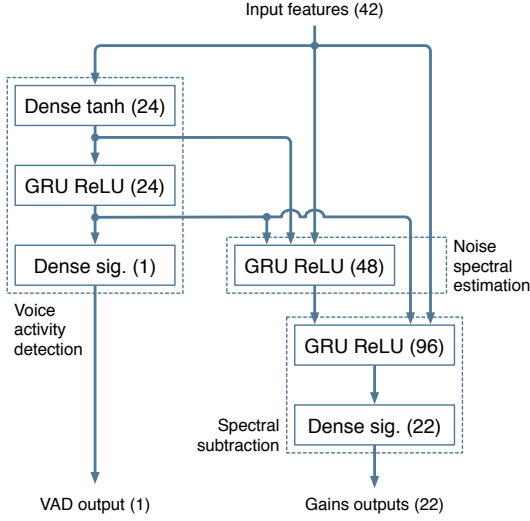
Figure 3. Architecture of the neural network, showing the feed-forward, fully connected (dense) layers and the recurrent layers, along with the activation function and the number of units for each layer.

96 units. We find that increasing the number of units does not significantly improve the quality of the noise suppression. However, the loss function and the way we construct the training data have a large impact on the final quality. We find that gated recurrent unit (GRU) [14] slightly outperform LSTM on this task, while also being simpler.

Despite the fact that it is not strictly necessary, the network includes a VAD output. The extra complexity cost is very small (24 additional weights) and it improves training by ensuring that the corresponding GRU indeed learns to discriminate speech from noise.

A. Training data

Since the ground truth for the gain requires both the noisy speech and the clean speech, the training data has to be constructed artificially by adding noise to clean speech data. For speech data, we use the McGill TSP speech database[1] (French and English) and the NTT Multi-Lingual Speech Database for Telephonometry[2] (21 languages). Various sources of noise are used, included computer fans, office, crowd, airplane, car, train, construction. The noise is mixed at different levels to produce a wide range of signal-to-noise ratios, including clean speech and noise-only segments.

Since we do not use cepstral mean normalization, we ensure robustness against variations in frequency responses by filtering each of the noise and speech signal independently using a second order filter of the form

$$H(z) = \frac{1 + r_1 z^{-1} + r_2 z^{-2}}{1 + r_3 z^{-1} + r_4 z^{-2}} \ , \tag{7}$$

where each of $r_1 \ldots r_4$ are random values uniformly distributed in the $\left[-\frac{3}{8}, \frac{3}{8}\right]$ range, which guarantees stability.

[1] http://www-mmsp.ece.mcgill.ca/Documents/Data/
[2] The 44.1 kHz audio CD tracks are used rather than the 16 kHz data files.

Robustness to the signal amplitude is achieved simply by varying the level of the mixed signal.

We have a total of 6 hours of speech and 4 hours of noise data, which we use to produce 140 hours of noisy speech from various combinations of gains and filters and by resampling them to frequencies between 40 kHz and 54 kHz.

B. Optimization process

The loss function used for training determines how the network weighs excessive attenuation versus insufficient attenuation when it cannot exactly determine the correct attenuation. Although it is common to use the binary cross-entropy function when optimizing for values in the $[0, 1]$ range, this does not produce good results for our gain because it does not match the perceived effect of the gain. For the gain estimate $\hat{g}$ and the ground truth g, we instead train with the loss function

$$L(g, \hat{g}) = (g^\gamma - \hat{g}^\gamma)^2 \ , \tag{8}$$

where the exponent γ is a perceptual parameter that controls how aggressively to suppress noise. Since $\lim_{\gamma \to 0} \frac{x^\gamma - 1}{\gamma} = \log(x)$, as $\gamma \to 0$, $L(g, \hat{g})$ minimizes the mean-squared log-energy, which would make the suppression too aggressive given the lack of a floor on the gain. In practice, the value $\gamma = 1/2$ provides a good trade-off and is equivalent to minimizing the mean squared error on the energy raised to the power $1/4$. Sometimes, there may be no noise and no speech in a particular band. This is common either when the input is silent or at high frequency when the signal is low-pass filtered. When that happens, the ground truth gain is explicitly marked as undefined and the loss function for that gain is ignored to avoid hurting the training process.

For the VAD output of the network, we use the standard cross-entropy loss function. Training is performed using the Keras[3] library with the Tensorflow[4] backend.

IV. COMPLEXITY ANALYSIS

To make it easy to deploy noise suppression algorithms, it is desirable to keep both the size and the complexity low. The size of the executable is dominated by the 87,503 weights needed to represent the 215 units in the neural networks. To keep the size as small as possible, the weights can be quantized to 8 bits with no loss of performance.

Since each weight is used exactly once per frame in a multiply-add operation the computational complexity of the neural network, each frame requires 174,000 floating-point operation (we count a multiply-add as two operations), so 17.4 Mflops for real-time use. The two FFTs per frame require around 7.5 Mflops and the pitch search (which operates at 12 kHz) requires around 10 Mflops. The total complexity of the algorithm is around 40 Mflops, which is comparable to that of a full-band speech coder.

A non-vectorized C implementation of the algorithm requires around 1.3% of a single x86 core (Haswell i7-4800MQ)

[3] https://keras.io/
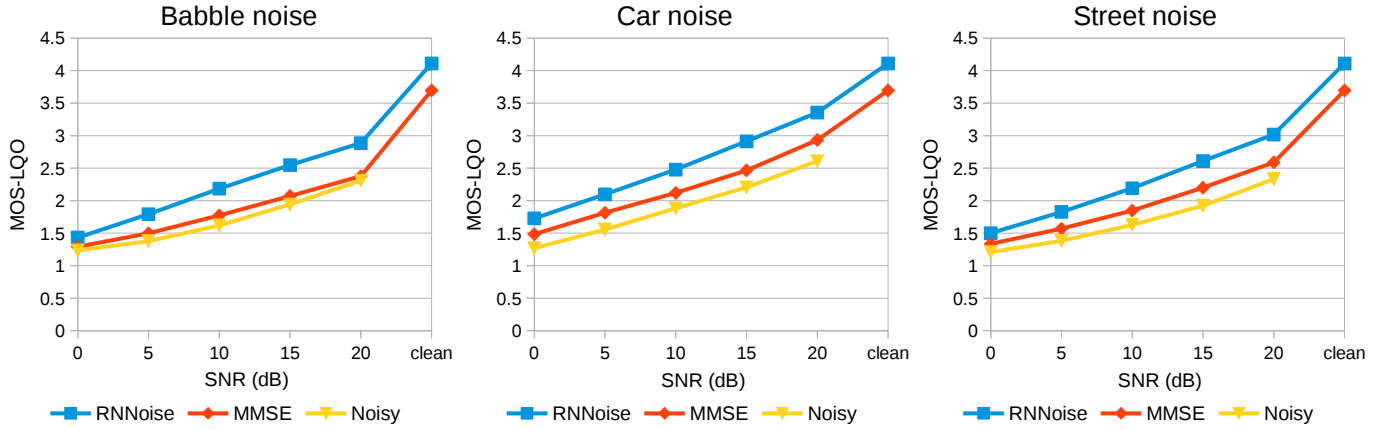[4] https://www.tensorflow.org/

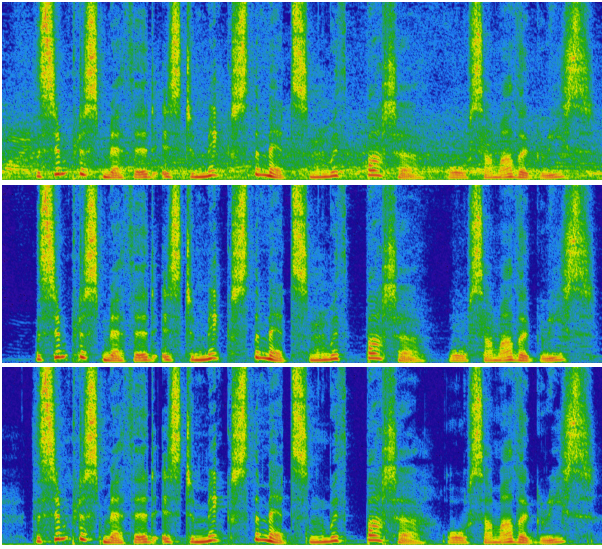Figure 4. PESQ MOS-LQO quality evaluation for babble, car, and street noise.

Figure 5. Example of noise suppression for babble noise at 15 dB SNR. Spectrogram of the noisy (top), denoised (middle), and clean (bottom) speech. For the sake of clarity, only the 0-12 kHz band is shown.

to perform 48 kHz noise suppression of a single channel. The real-time complexity of the same floating-point code on a 1.2 GHz ARM Cortex-A53 core (Raspberry Pi 3) is 14%.

V. RESULTS

We test the quality of the noise suppression using speech and noise data not used in the training set. We compare it to the MMSE-based noise suppressor in the SpeexDSP library[5]. Although the noise suppression operates at 48 kHz, the output has to be resampled to 16 kHz due to the limitations of wideband PESQ [15]. The objective results in Fig. 4 show a significant improvement in quality from the use of deep learning, especially for non-stationary noise types. The improvement is confirmed by casual listening of the samples. Fig. 5 shows the effect of the noise suppression on an example.

A demonstration of the proposed system is available at https://people.xiph.org/~jm/demo/rnnoise/ including an interactive real-time Javascript implementation. The software described in this paper can be obtained at https://github.com/xiph/rnnoise/ and the results were produced using commit hash cdf196b.

VI. CONCLUSION

This paper demonstrates a noise suppression approach that combines DSP-based techniques with deep learning. By using deep learning only for the aspects of noise suppression that are hard to tune, the problem is simplified to computing only 22 ideal critical band gains, which can be done efficiently using few units. The coarse resolution of the bands is then addressed by using a simple pitch filter. The low resulting complexity makes the approach suitable for use in video-conferencing systems. We demonstrate that the quality is significantly higher than that of a pure signal processing-based approach.

We believe that quality can further be improved by using a more advanced pitch filter (e.g. an IIR version). We also believe that the technique can be easily be extended to residual echo suppression, for example by adding the cepstrum of the far end signal or the filtered far-end signal to the input of the neural network.

REFERENCES

[1] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. on acoustics, speech, and signal processing*, vol. 27, no. 2, pp. 113–120, 1979.

[2] H.-G. Hirsch and C. Ehrlicher, "Noise estimation techniques for robust speech recognition," in *Proc. ICASSP*, 1995, vol. 1, pp. 153–156.

[3] T. Gerkmann and R.C. Hendriks, "Unbiased MMSE-based noise power estimation with low complexity and low tracking delay," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1383–1393, 2012.

[4] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 443–445, 1985.

[5] A. Maas, Q.V. Le, T.M. O'Neil, O. Vinyals, P. Nguyen, and A.Y. Ng, "Recurrent neural networks for noise reduction in robust ASR," in *Proc. INTERSPEECH*, 2012.

[6] D. Liu, P. Smaragdis, and M. Kim, "Experiments on deep learning for speech denoising," in *Proc. Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[7] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 23, no. 1, pp. 7–19, 2015.

[8] A. Narayanan and D. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition," in *Proc. ICASSP*, 2013, pp. 7092–7096.

[9] C. Montgomery, "Vorbis I specification," 2004.

[10] J. Princen and A. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation," *IEEE Tran. on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1153–1161, 1986.

[11] B.C.J. Moore, *An introduction to the psychology of hearing*, Brill, 2012.

[12] J.-M. Valin, G. Maxwell, T. B. Terriberry, and K. Vos, "High-quality, low-delay music coding in the Opus codec," in *Proc. 135th AES Convention*, 2013.

[13] Juin-Hwey Chen and Allen Gersho, "Adaptive postfiltering for quality enhancement of coded speech," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 59–71, 1995.

[14] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proc. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[15] ITU-T, *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*, 2001.