Experiments on Deep Learning for Speech Denoising

Ding Liu[1], *Paris Smaragdis*[1,2], *Minje Kim*[1]

[1]University of Illinois at Urbana-Champaign, USA
[2]Adobe Research, USA

Abstract

In this paper we present some experiments using a deep learning model for speech denoising. We propose a very lightweight procedure that can predict clean speech spectra when presented with noisy speech inputs, and we show how various parameter choices impact the quality of the denoised signal. Through our experiments we conclude that such a structure can perform better than some comparable single-channel approaches and that it is able to generalize well across various speakers, noise types and signal-to-noise ratios.

Index Terms: Speech denoising, Deep Learning, Neural networks, Source Separation

1. Introduction

The goal of speech denoising is to produce noise-free speech signals from noisy recordings, while improving the perceived quality of the speech component and increasing its intelligibility. Speech denoising can be utilized in various applications where we experience the presence of background noise in communications. A number of techniques have been proposed based on different assumptions on the signal and noise characteristics, including spectral subtraction [1] statistical model-based estimation [2], Wiener filtering [3], subspace method [4] and non-negative matrix factorization (NMF) [5]. In this paper we introduce a lightweight learning-based approach to remove noise from single-channel recordings using a deep neural network structure.

Neural networks as a non-linear filter have been applied to this problem in the past, for example the early work by [6] utilizing shallow neural networks (SNNs) for speech denoising. However, at that time constraints in computational power and size of training data resulted in relatively small neural network implementations that limited denoising performance.

Over the last few years, the development of computer hardware and advanced machine learning algorithms enabled people to increase the depth and width of neural networks. The deep neural networks (DNNs) have achieved many state-of-the-art results in the field of speech recognition [7] and speech separation [8]. DNNs containing multiple hidden layers of non-linearity have shown great potential to better capture the complex relationships between noisy and clean utterances across various speakers, noise types and noise levels. More recently, Xu et al. [9] proposed a regression-based speech enhancement framework of DNNs using restricted Boltzmann machines (RBMs) for pre-training.

In this paper we explore the use of DNNs for speech denoising, and propose a simpler training and denoising procedure that does not necessitate RBM pre-training or complex recurrent structures. We use a DNN that operates on the spectral domain of speech signals, and predicts clean speech spectra when presented with noisy input spectra. A series of experiments is conducted to compare the denoising performance under different parameter settings. Our results show that our simplified approach can perform better than other popular supervised single-channel denoising approaches and that it results in a very efficient processing model which forgoes computationally costly estimation steps.

2. Neural Networks for Spectral Denoising

In the following sections we introduce our model's structure, some domain-specific choices that we make, and a training procedure optimized for this task.

2.1. Network Structure

The core concept in this paper is to compute a regression between a noisy signal frame and a clean signal frame in the frequency domain. To do so we start with the obvious choice of using frames from a magnitude short-time Fourier transform (STFT). Using these features allows us to abstract many of the phase uncertainties and to focus on "turning off" parts of the input spectral frames that are purely noise [6].

More precisely, for a speech signal $s(t)$ and a noise signal $n(t)$ we construct a corresponding mixture signal $m(t) = s(t) + n(t)$. We compute the STFTs of the above time series to obtain the vectors $\mathbf{s}_t, \mathbf{n}_t$ and $\mathbf{m}_t$, which are the spectral frames corresponding to time t (each element of these vectors corresponds to a frequency bin). These vectors will constitute our training data set, with $\mathbf{m}_t$ being the input and its corresponding $\mathbf{s}_t$ being the target output.

We then proceed to design a neural network with L layers which would output a spectral frame prediction $\mathbf{y}_t$ when it is presented with $\|\mathbf{m}_t\|$. This is akin to a Denoising Autoencoder (DAE) [10], although in this case we do not care to find an efficient hidden representation, but instead we care to predict the spectra of a clean signal when provided with the spectra of a noisy signal. The runtime denoising process is defined by:

$$\mathbf{h}_t^{(l)} = f_l\left(\mathbf{W}^{(l)} \cdot \mathbf{h}_t^{(l-1)} + \mathbf{b}^{(l)}\right) \quad (1)$$

with l signifying the layer index (from 1 to L), and with $\mathbf{h}_t^{(0)} = \|\mathbf{m}_t\|$ and $\mathbf{y}_t = \mathbf{h}_t^{(L)}$. The function $f_l(\cdot)$ is known as the activation function and can take various forms depending on our goals, but it is traditionally a sigmoid or some piecewise linear function. We will explore this selection in a later section. Likewise the number of layers L can range from 1 (which forms a shallow network), or as many as we deem necessary (which comes with a higher computational burden and the need for more training data).

For $L = 1$ and $f_l(\cdot)$ being the identity function this model collapses to a linear regression, whereas when using non-linear $f_l(\cdot)$'s and multiple layers we perform a deep non-linear regression (or a regression deep neural network).

2.2. Training Procedure

The parameters that need to be estimated in order to obtain a functioning system are the set of $\mathbf{W}^{(l)}$ matrices and $\mathbf{b}^{(l)}$ vectors, known as the layer weights and biases respectively. Fixed parameters that we will not learn include the number of layers L and the choice of activation functions $f_l(\cdot)$. In order to perform training we need to specify a cost function between the network predictions and the target outputs which we will need to optimize, and that will provide a means to see how well our model has adapted to the training data.

For the activation function the most common choices are the hyperbolic tangent and the logistic sigmoid function. However we note that the outputs that we wish to predict are spectral magnitude values which would lie in the interval $[0, \infty)$. This means that we should prefer an activation function that produces outputs in that interval. A popular choice that satisfies this preference is the rectified linear activation, which is defined as $y = \sup\{x, 0\}$ i.e. the maximum between the input and 0. In our experience, however, this is a particularly difficult function to work with since it exhibits a zero derivative for negative values and is very likely to result in nodes that get "stuck" with a zero output once they reach that state. Instead we use a modified version which is defined as:

$$f(x) = \begin{cases} x & \text{if } x \geq \epsilon \\ \frac{-\epsilon}{x-1-\epsilon} & \text{if } x < \epsilon \end{cases}$$

where ϵ is a sufficiently small number (in our simulations set to 10^{-5}). This modification introduces a slight ramp starting from $-\infty$ to ϵ which guarantees that the derivative will point (albeit weakly) towards positive values and will provide a way to escape a zero state once a node is in it.

For the cost function we select the mean squared error (MSE) between the target and predicted vectors: $E \propto \|\mathbf{y}_t - \|\mathbf{s}_t\|\|^2$. Although a choice such as the KL divergence or the Itakura-Saito divergence would have been more appropriate for measuring differences between spectra, in our experiments we find them to ultimately perform worse than the MSE.

Once the above network characteristics have been specified we can use a variety of methods to estimate the model parameters. Traditional choices include the backpropagation algorithm, as well as more sophisticated procedures such as conjugate gradient methods and optimization approaches such as Levenberg-Marquardt [11]. Additionally, there is a trend towards including a pre-training step using an RBM analogy for each layer [12]. In our experiments for this specific task, we find many of the sophisticated approaches to be either numerically unstable, computationally too expensive, or plainly redundant. We obtain the most rapid and reliable convergence behavior using the resilient backpropagation algorithm [13]. Combined with the use of the modified activation function that we present above, it requires no pre-training and converges in roughly the same number of iterations as conjugate gradient algorithms with far fewer computational requirements. The initial parameter values are set using the Nguyen-Widrow procedure [14]. For most of the experiments we train our models for 1,000 iterations which are usually sufficient to achieve convergence. The details regarding the training data are discussed in the experimental results section.

2.3. Extracting the Denoised Signal

After training a model, the denoising is performed as follows: the magnitude spectral frames from noisy speech signals are extracted and presented as inputs. If the model is properly trained we obtain a prediction of the clean signal's magnitude spectrum for each noisy spectrum that we analyze. In order to invert that magnitude spectrum back to the time domain, we apply the phase of the mixture spectrum on it and we use the inverse STFT with overlap-add to synthesize the denoised signal in the time domain. For all our experiments we use a square-root Hann window for both the analysis and synthesis transforms, and a hop size of 25% of the Fourier window length.

2.4. Dealing with Gain

One potential problem with this scheme is that this network might not be able to extrapolate when presented with data at significantly large scales (e.g. 10x louder). When using large data sets there is a high probability that we will see enough spectra at various low gains to adequately perform regression at lower scales, but we will not observe spectra louder than some threshold which means that we will not be able to denoise very loud signals. One approach is to standardize the gain of the involved spectra to lie inside a specific range, but we can instead employ some simple modifications to help us extrapolate better.

In order to do so we perform the following steps. We first normalize all the input and output spectra to have the same ℓ_1-norm (we arbitrarily choose unit norm). In the training process we add one more output node that is trained to predict the output gain of the speech signal. The target output gain values are also normalized to have unit variance over an utterance in order to impose invariance on the scale of the desired output signal. With this modification, in order to obtain the spectrum of the denoised signal we would have to multiply the output of that gain node with the speech spectrum predicted from all the other nodes. Because of the normalization on the predicted gain we will not recover the clean input signal with the exact gain, but rather a denoised signal that has roughly the same amplitude modulation with a constant scaling factor. In the next section we show how this method compares to simply training on unnormalized spectra.

3. Experimental Results

We now present the results of experiments that explore the effects of relevant signal and network parameters, as well as the degradation in performance when the training data set does not adequately represent the testing data.

The experiments are set up using the following recipe. We use one hundred utterances from the TIMIT database, spanning ten different speakers. We also maintain a set of five noises specified as: Airport, Train, Subway, Babble and Drill. We then generate a number of noisy speech recordings by selecting random subsets of noise and overlaying them with speech signals. While constructing the noisy mixtures we also specify the signal to noise ratio for each recording. Once we complete the generation of the noisy signals we split them into a training set and a test set.

During the denoising process we can specify multiple parameters that have a direct effect on separation quality and are linked to the network's structure. In this paper we present the subset that we find to be most important. These include the number of input nodes, the number of hidden layers and the number of their nodes, the activation functions, and the number of prior input frames to take into account.

Of course the number of parameters is quite large and considering all the possible combinations is an intractable task.

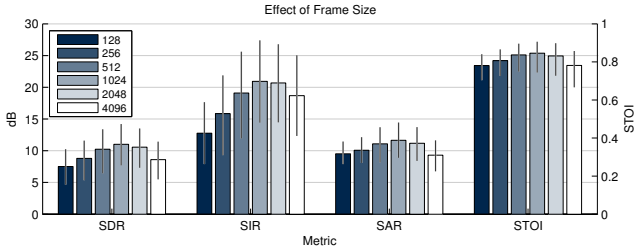

Figure 1: *Comparing different input FFT sizes we see that for speech signals sampled at 16kHz we obtain the optimal results with 1024pts. As with all figures in this paper, the bars show average values and the vertical lines on the bars denote minimum and maximum observed values from our experiments.*

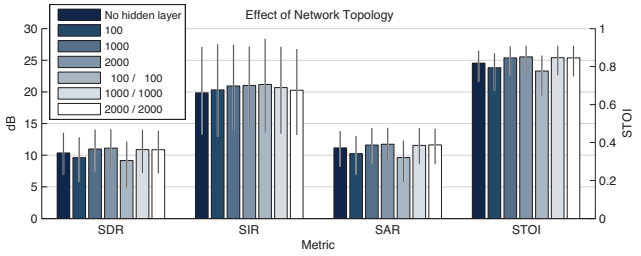

Figure 2: *Comparing different network structures we see that a single hidden layer with 2000 units seems to perform best. Entries corresponding to a single legend number denote a single hidden layer with that many hidden units. Entries corresponding to two legend numbers denote a two hidden layer network with the two numbers being the units in the first and second hidden layer, respectively.*

In the following experiments we perform single parameter searches while keeping the rest of the parameters fixed in a set of sensible choices according to our observations. The fixed parameters are: input frame size 1024pts, a single hidden layer with 2000 units, the rectified linear activation with the modification described above, 0dB SNR inputs, no input normalization, and no temporal memory.

For all parameter sweeps we show the resulting signal to distortion ratio (SDR), signal to interference ratio (SIR) and signal to artifacts ratio (SAR) as computed from the BSS-EVAL toolbox [15]. We additionally compute the short-time objective intelligibility measure (STOI) which is a quantitative estimate of the intelligibility of the denoised speech [16]. For all these measures higher values are better.

3.1. Network Structure

In this section we present the effects of the network's structure on performance. We focus on four parameters that we find to be the most crucial, namely input window size, number of layers, activation function and temporal memory.

The number of input nodes is directly related to the size of the analysis window that we use, which is the same as the size of the FFT that we use to transform the time domain data to the frequency domain. In Figure 1 we show the effects of different window sizes. We see that a window of about 64 ms (1024pts) produces the best result.

Another important parameter is that of the depth and width of the network, i.e. the number of hidden layers and their corresponding nodes. In Figure 2 we show the results over various settings ranging from a simple shallow network to a two-hidden layer network with 2000 nodes per layer. We note that with more units we tend to see an increase in the SIR, but that this trend stops after a while. It is not clear if this is an effect that relates to the number of training data points that we use or not. Regardless the SDR, SAR and STOI seem to require more hidden layers with more units. Consolidating both observations we note that a single hidden layer with 2,000 units is optimal.

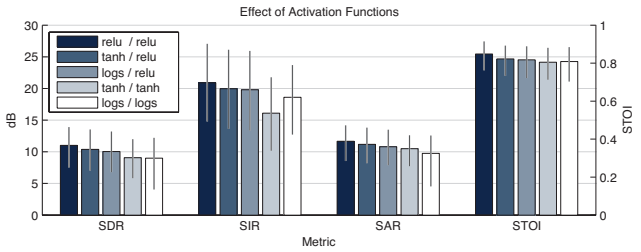

Figure 3: *Comparing different activation functions we see that the rectified linear activation outperforms other common functions. The legend entries show the activation function for the hidden and the output layer, with "relu" being the rectified linear, "tanh" being the hyperbolic tangent and "logs" being the logistic sigmoid.*

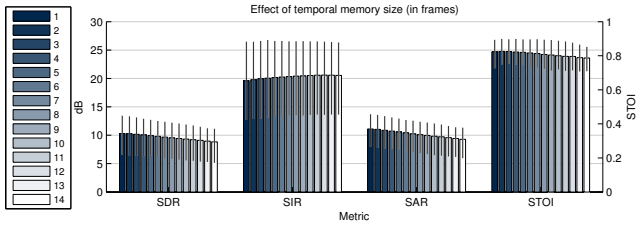

Figure 4: *Using a convolutive form that takes into account prior input frames, we note that although SIR performance increases as we include more past frames there is an overall degradation in quality after more than two frames.*

We also examine the effect of various activation functions with the results shown in Figure 3. The ones that we consider are the rectified linear activation (with the modifications described above), the hyperbolic tangent and the logistic sigmoid function. For all cases it seems that the modified rectified linear activation is consistently the best performer.

Finally we examine the effects of a convolutive structure on the input as shown in Figure 4. We do so using a model that receives as input the current analysis window as well as an arbitrary number of past windows. The number of past windows ranges from 0 to 14 in our experiments. We observe a familiar pattern in the measured results, where the SIR improves at the expense of a diminishing SDR/SAR/STOI. Overall we conclude that the input of two consecutive frames is a good choice, although even a simple memoryless model would perform reasonably well enough.

3.2. Robustness to Variations

In order to evaluate the robustness of this model, we test it under a variety of situations in which it is presented with unseen data, such as unseen SNRs, speakers and noise types.

In Figure 5 we show the robustness of this model under var-

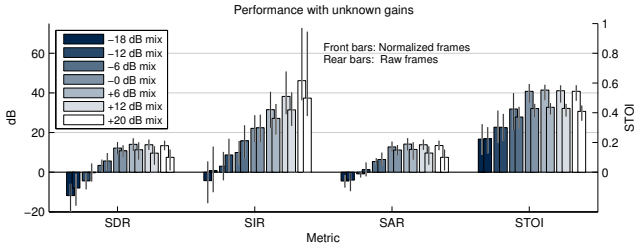

Figure 5: *Using multiple SNR inputs and testing on a network that is trained on 0dB SNR. Note that the results are absolute, i.e. we do not show the improvement. All results are shown using pairs of bars. The left/back bars in each pair show the results when we train on raw data, and the right/front bars show the results when we do the gain prediction.*

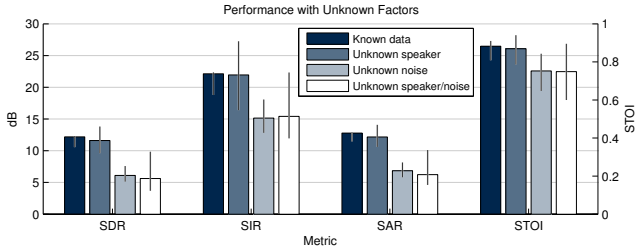

Figure 6: *In this figure we compare the performance of our network when used on data that is not represented in training. We show the results of separation with known speakers and noise, with unseen speakers, with unseen noise, and with unseen speakers and noise.*

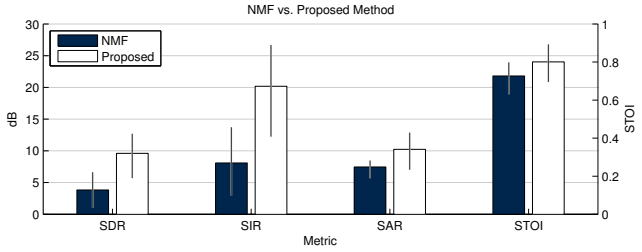

Figure 7: *Comparison of the proposed approach with NMF-based denoising.*

ious SNRs. The model is trained on 0dB SNR mixtures and it is evaluated on mixtures ranging from 20 dB SNR to -18dB SNR. We additionally test both the method to train on the raw input data and the method using the gain prediction model described above. In Figure 5 these two methods are compared with the use of the front and back bars. Note that the shown values are absolute, not the improvement from the input mixture. As we see, for positive SNRs we get a much improved SIR and a relatively constant SDR/SIR/STOI, and training on the raw inputs seems to work better. For negative SNRs we still get an improvement although it is not as drastic as before. We also note that in these cases training with gain prediction tends to perform better.

Next we evaluate this method's robustness to data that is unseen in the training process. These tests provide a glimpse of how well we can expect this approach to work when applied on noise and speakers on which it is not trained. We perform three experiments for this, one where the testing noise is not seen in training, one where the testing speaker is not seen in training, and one where both the testing noise and the testing speaker are not seen in training. For the unseen noise case we train the model on mixtures with Babble, Airport, Train and Subway noises, and evaluate it on mixtures that include a Drill noise (which is significantly different from the training noises in both spectral and temporal structure). For the unknown speaker case we simply hold out from the training data some of the speakers, and for the case where both the noise and the speaker are unseen we use a combination of the above. The results of these experiments are shown in Figure 6. For the case where the speaker is unknown we see only a mild degradation in performance, which means that this approach can be easily used in speaker variant situations. With the unseen noise we observe a larger degradation in results, which is expected due to the drastically different nature of the noise type. Even then, the result is still good enough as compared to other single-channel denoising approaches. The result of the case where both the noise and the speaker are unknown seems to be at the same level as that of the case of the unseen noise, which once again reaffirms our conclusion that this approach is very good at generalizing across speakers.

4. Conclusions

To conclude we present one more plot that shows how this approach compares to another popular supervised single-channel denoising approach. In Figure 7 we compare our performance to a non-negative matrix factorization (NMF) model trained on the speakers and noise at hand [5]. For the NMF model we use what we find to be the optimal number of basis functions for this task. It is clear that our proposed method significantly outperforms this approach.

Based on the above experiments we can form a series of conclusions. Primarily we see that this approach is a viable one, being adequately robust to unseen mixing situations (both with gains and types of sources). We also see that a deep or convolutive structure is not crucial, although it does offer a minor performance advantage. In terms of activation functions we note that the rectified linear activation seems to perform the best. Our proposed approach provides a very efficient runtime denoising process which is comprised of only a linear transform on the size of the input frame followed by a max operation. This brings our approach in the same level of computational complexity as spectral subtraction, while offering a significant advantage in denoising performance. Unlike methods such as NMF-based denoising there is no estimation performed at runtime which makes for a significantly more lightweight process.

Of course our experiments are not exhaustive, but they do provide some guidelines on what structure to use to achieve good denoising results. We expect that with further experiments measuring many more of the available options, in both training and post-processing, we can achieve even better performance.

5. Acknowledgement

The authors would like to acknowledge NVIDIA's kind support in providing the computing resources for these experiments.

6. References

[1] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 27, no. 2, pp. 113–120, 1979.

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

[3] P. Scalart *et al.*, "Speech enhancement based on a priori signal to noise estimation," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 2. IEEE, 1996, pp. 629–632.

[4] Y. Ephraim and H. L. Van Trees, "A signal subspace approach for speech enhancement," *Speech and Audio Processing, IEEE Transactions on*, vol. 3, no. 4, pp. 251–266, 1995.

[5] K. W. Wilson, B. Raj, P. Smaragdis, and A. Divakaran, "Speech denoising using nonnegative matrix factorization with priors." in *ICASSP*, 2008, pp. 4029–4032.

[6] E. A. Wan and A. T. Nelson, "Networks for speech enhancement," *Handbook of neural networks for speech processing. Artech House, Boston, USA*, 1999.

[7] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[8] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *ICASSP*, 2014 In Press.

[9] Y. Xu, J. Du, L. Dai, and C. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Processing Letters*, vol. 21, no. 1, 2014.

[10] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[11] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin, *Neural networks and learning machines*. Pearson Education Upper Saddle River, 2009, vol. 3.

[12] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[13] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE International Conference on Neural Networks*, 1993, pp. 586–591.

[14] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, June 1990, pp. 21–26 vol.3.

[15] C. Févotte, R. Gribonval, and E. Vincent, "Bss eval, a toolbox for performance measurement in (blind) source separation." [Online]. Available: http://bass-db.gforge.inria.fr/bss_eval

[16] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 7, pp. 2125–2136, 2011.