Speech Enhancement Using Deep Autoencoder

Mingchao Liang
University of California San diego
9500 Gilman Dr, La Jolla, CA 92093
m3liang@eng.ucsd.edu

Weinan Li
University of California San diego
9500 Gilman Dr, La Jolla, CA 92093
w3li@eng.ucsd.edu

Arda Cankat Bati
University of California San diego
9500 Gilman Dr, La Jolla, CA 92093
abati@eng.ucsd.edu

Marjan Emadi
University of California San diego
9500 Gilman Dr, La Jolla, CA 92093
memadi@eng.ucsd.edu

Abstract

In this paper we study the usage of Deep Autoencoders (DAE) for noise reduction of noisy speech recordings. We train our network using both clean speech and noisy speech recordings. The noisy speech recordings are constructed using an additive noise model. The trained DAE network is used to reduce background noise from the noisy recordings. For the evaluation of the results, objective methods such as Segmental Signal to Noise Ratio (SSNR) and Perceptual Evaluation of Speech Quality (PESQ) are used. We also compare our model's performance to two traditional methods, namely Wavelet Shrinkage and logarithm of the Minimum of Mean Square Error,Log-MMSE. Experimental results reveal that the implemented model is able to successfully reduce noise, for different noise types and SNR values. Noise was also successfully reduced for noise types previously unknown by the network. In the results section, we show the performance comparisons with different network parameters architectures. Finally, comparisons we made to traditional methods show that the DAE network performs better regarding SSNR values, while it is not ideal considering PESQ evaluation.

1. Introduction

The problem of enhancing noisy speech recorded by a single microphone has attracted much research effort in speech communications. The purpose of our research is to extract clean speech out of noisy recordings. The noise may originate from different environments such as restaurants, streets, airports etc. It may also be caused by sources such as recording equipment or communications media. Our main aim in the enhancement of the noisy speech is to increase the quality of speech for human perception. It can also be beneficial for voice recognition frameworks, although this is not our main concern.

In this paper, we will focus on human perception, using objective methods to evaluate the performance of our trained model. Noise filtering methods such as Log-MMSE, Wavelet Shrinkage have been traditionally used for the purpose of enhancing noisy recordings. We hope to show that utilizing Deep Neural Networks will provide noticeably better results than these traditional methods. We will use objective evaluation models such as Segmental Signal to Noise Ratio (SSNR) and Perceptual Evaluation of Speech Quality (PESQ).

We are using the Deep Autoencoder architecture for our Deep Neural networks, which is explained in the Autoencoder section. For our train and test sets, we will use the speech recordings from TIMIT Corpus of read speech. The details of this dataset will be mentioned in the Dataset Generation section of our paper.

For the methodology of our project, it is important to note that we are inspired by the work of Xu et al. [1] They use an approach based on Restricted Boltzman Machines. While we decided to take the DAE approach, we used similar guidilines to the ones they provide, specifically for the dataset generation and evaluation methods.

2. Dataset Generation

2.1. Overview

For the purposes of this paper, we built our dataset from two sources: clean speech [2][3](TIMIT Corpus), and noise recordings Signal Processing Information Base (SPIB) noise dataset [4]. We will briefly explain the contents of the datasets themselves in the next subsection. We are using an Additive Noise Model to create our dataset. Different noise sources from SPIB are added to the clean

speech files in different Signal to Noise Ration (SNR) levels. The details of this process will be explained in the subsection Noisy Speech Generation.

2.2. TIMIT and SPIB Datasets

In this section we briefly describe the contents of the TIMIT and SPIB Datasets.

2.2.1 TIMIT Corpus

TIMIT Dataset: TIMIT dataset contains 10 sentences spoken by 630 different speakers (in total 6300 sentences.) The speakers are categorized into 8 dialects. The male / female distribution of the data is 70%, 30%. The train and test set distribution is 4620 and 1680 sentences respectively The sound files are in wave (.wav) file format with 512 kbit/s. We down sample to 128 kbit/s for faster training, and to make them compatible with SPIB noise dataset.

2.2.2 SPIB Noise Dataset

SPIB Noise Dataset has 15 recordings for different noise types. We are using 6 of these noise types for our paper. Each recording is 3 minutes 55 seconds long. The sound files are in wave file format with 128 kbit/s .

2.3. Noisy Speech Generation

The noisy versions of the TIMIT dataset are created by adding the noise recordings from SPIB over the clean speech files. Many traditional speech enhancement methods assume that the noise and the speech are independent. However the DNN model we use assumes a non-linear mapping function without using any special assumptions about the relation between the noise and original recordings. Therefore, the methodology we use may be generalized to different types of noises such as convolutional noise. However, further experiments should be done to confirm this, which is out of scope for our paper.

We use 4 noise types to generate our train set (alarm, babble, destroyer-engine and white). For the test set we use the same noise types plus 2 others (pink, volvo). With the different noise types we aim to evaluate the success of our trained network with unseen noise types (which will be commonplace in practice).

We add the noise to our speech files in different SNR values: -5dB, 0dB, 5dB, 10dB, 15dB, 20dB. (From the most noisy to least noisy additions). In total, we have for train set: 4 noise types x 6 SNRS = 24 combinations, for test set: 6 noise types x 6 SNRS = 36 combinations. Therefore our train set grows by x24 and the test set by x36 in size. We have decided to randomly select 200 recordings from the original test set, and use these for our tests purposes. We do this to speed up the process and we believe that 200 speech recordings are enough to evaluate the models.

The speech recordings are typically around 2-5 seconds long each while we have 3:55 long noise recordings. While adding the noise to clean speech files we randomly select a slice from the noisy file and add it to the clean recording.

3. Preprocessing

Steady state sounds, like vowels are produced by periodic excitation of a linear system. Speech is a time varying signal and it needs more sophisticated analysis to reflect time varying properties. We use Short Time Fourier Transform (STFT), to change the domain of the signal from time to frequency. Here the number of frequency components is 128. We use the Discrete STFT,as the following formula:

$$\begin{aligned} STFT\{x[n]\}(m,\omega) &= X(m,\omega) \\ &= \sum_{n=-\infty}^{\infty} x[n]w[n-m]\exp\left(-j\omega n\right) \end{aligned}$$

Where x is the signal to be transformed,w is the window function and ω is the angular frequency.

Sounds are usually in high frequencies after doing the STFT, and humans is unable to hear very large amount of frequencies, that is why we use the log power to change the frequencies to dB.

The data was also normalized to zero mean and unit variance per-feature-based during the preprocessing.

4. Autoencoder

Autoencoder is a widely used neural network architecture for feature extraction. Similar to PCA, autoencoder is unsupervised, with its input being its target as well. PCA finds a lower dimensional hyperplane to project the data onto, preserving most of the variance. The autoencoder can find more complex spaces to project the data onto, and therefore it is usually more successful in non linear contexts, such as speech.

A typical autoencoder consists of two parts, an encoder and a decoder. The encoder maps the input to a representation space, encoding/compressing the input according to a non linear function. The decoder takes this encoded form of the data and tries to reconstruct the original input from it. It builds its own representation space, which is again a non-linear function. Ideally the reconstructed input is the same as the original input. The encoder part alone is usually used as a feature extractor of the input.

Denoising autoencoder [5], an improved version of autoencoder, is designed to achieve robust feature extraction with dropout. In an autoencoder, when there are are more nodes in the hidden layer than there are inputs, the autoencoder is at the risk of learning Identity Function (also called

Null Function). This means just learning the function output = input, rendering the network useless. Dropouts are used to prevent this from happening. Instead of applying dropout in the hidden layer, denoising autoencoder applies dropout in the input layer by randomly masking part of the input. The dropout removes some of the connection between the input features, hence better extracts the overall pattern behind, instead of memorizing it.

Our model, instead of using autoencoder to do feature extraction, takes the noisy spectrogram as the input and the clean spectrogram as the target. The diagram is shown as follow:

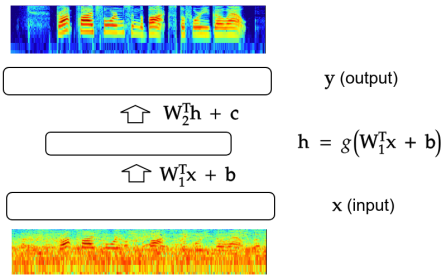

Figure 1. Diagram of the autoencoder.

The function g here is the non-linear activation in the hidden layer. In our study, g is leaky ReLU, which is a piece-wise linear function:

$$g(x) = \begin{cases} x, & x \geq 0 \\ ax, & x < 0 \end{cases}$$

where the a is a positive parameter that can be learned through backpropagation. The $\mathbf{W}_1$ and $\mathbf{W}_2$ are the encoder and decoder matrix respectively. $\mathbf{b}$ and $\mathbf{b}$ are the biases of the input and output layers. The parameters $\mathbf{\Phi} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}, \mathbf{c}\}$ are determined by minimizing the mean square error (MSE) between the output $\mathbf{y}$ and the target $\mathbf{t}$:

$$J(\mathbf{\Phi}) = \sum_i \|\mathbf{t}_i - \mathbf{y}_i\|_2^2$$

$$\mathbf{\Phi}^* = \min_{\mathbf{\Phi}} J(\mathbf{\Phi})$$

A deep autoencoder (DAE) can be built by stacking several autoencoders. We will discuss the effect of the number of stacked autoencoders in the experiments section.

Below are the definitions we are using for tied / untied and symmetrical / non symmetrical autoencoders.

For each single autoencoder: Symmetric structure means The weights of of the decoder is the transpose of the weights of the encoder (i.e. $\mathbf{W}_1 = \mathbf{W}_2^T$). Non Symmetric structure means the encoder and the decoder do not share weights.

For the cascade of autoencoders in the deep autoencoders structure, we have: Tied structure, all the cascaded autoencoders have shared weights (i.e. every autoencoder uses the same $\mathbf{W}_1$ and $\mathbf{W}_2$). Untied structure, the cascaded autoencoders' weights are not shared with each other.

5. Experiments

In this section we first mention our training setup and execution. We then describe how we use different parameters to evaluate our model, and try to find the best combinations.

5.1. Training

5.1.1 Training Process

We use the ADAM algorithm [6] to train our model, which is an extension of the stochastic gradient descent. After being introduced in 2014, it quickly become a common place algorithm to use in Deep Learning related applications. We use the algorithm with mini batches that have 512 samples . Each sample is a slice from the log power spectrogram of noisy speech data (as input) and clean speech data (as target). In most of our experiments, we use 11 frames for each sample, (we also use 15 and 21 as alternative frame sizes).

5.1.2 Choosing Parameters

For the evaluation of our results with different network architectures and parameters in a limited time frame, we decided to set two baselines, from which we will deviate to compare performances. These baselines are as follows:

Baseline 1:

- Non Symmetric autoencoder
- Number of stacked autoencoders = 3
- Hidden Layer Size = 500
- Number of frames for each sample = 11
- Learning rate = 0.001
- Train Data speech recordings count = 800

Baseline 2 is the same as Baseline 1, but with symmetric autoencoders. We use these two baselines as the starting point, and deviate from the baseline by changing individual parameters, and training a new network. For each deviation, the only difference from the baseline is a change in a single parameter. We then compare the performance of the deviation to the baseline, and decide if the change can be considered successful. Below we show the deviations we tried from Baseline 1 and Baseline 2:

Deviations from Baseline 1

- Number of layers: 2,5,7
- Learning Rate: 0.01, 0.005

- Number of Frames: 15, 21

Deviations from Baseline 2

- Hidden layer size: 300, 800
- Number of speech recordings used in train set: 400, 3200

5.2. Testing

During the testing stage, we store the phase of the spectrogram (with complex components) of the noisy raw audio before converting it to log power spectrogram. After the we get the restored log power spectrogram, we combined it with the stored phase to form a restore spectrogram (with complex components). Then we do inverse STFT to obtain the restored raw audio.

6. Results

For the results section, we will consider each parameter seperately, in its own subsection. The compared models in each subsection were trained with the same number of epochs.

6.1. Example Training Loss Graph

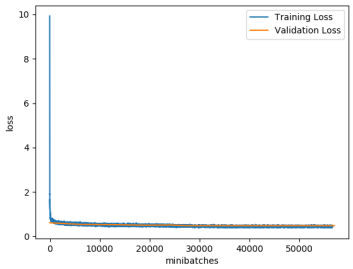

Figure 2. Example Train Loss Graph from the training process.

6.2. Number of Cascaded Autoencoders

The results in this section show that building too deep autoencoders do not lead to better results. In fact the ideal layer size seems to be either 2 or 3 for the purposes of our experiment. For SSNR values 2 layered DAE performs better for more noisy, low SNR examples, while 3 Layered DAE performs better for less noisy, higher SNR examples. The PESQ [7]results, on the other hand strongly favor DAE with 2 layers. As there is no clear better layer count from our results, different evaluation methods could be used to make a more concrete statement about number of layers.

Table 1. Effect of the number of stacked autoencoders (AEs) on SSNR(dB)

# AEs	-5dB	0dB	5dB	10dB	15dB	20dB
7	3.58	5.31	6.70	7.54	7.91	9.61
5	3.80	5.83	7.56	8.69	9.26	9.41
3	3.82	5.94	7.96	**9.48**	**10.28**	**10.51**
2	**4.09**	**6.24**	**8.15**	**9.48**	10.18	10.31

Table 2. Effect of the number of stacked autoencoders (AEs) on PESQ

# AEs	-5dB	0dB	5dB	10dB	15dB	20dB
7	1.96	2.19	2.32	2.63	2.77	2.88
5	**2.07**	2.18	2.43	2.65	2.82	2.91
3	2.02	2.21	2.39	2.59	2.75	2.94
2	2.05	**2.26**	**2.49**	**2.71**	**2.94**	**3.09**

6.3. Size of Hidden Units

Comparing the size of hidden units, we see a significant change from 300 to 500. From 500 to 800, the results are still mostly improving, however it seems that we are approaching a maximum, ideal hidden unit size with 800. Trying several higher sized hidden units could be valid approach, however we concluded our experiments at this stage due to time limitations.

Table 3. Effect of the size of hidden units on SSNR(dB)

# Units	-5dB	0dB	5dB	10dB	15dB	20dB
300	3.89	5.88	7.77	9.08	9.72	9.84
500	4.23	6.40	8.51	10.08	10.85	11.01
800	**4.24**	**6.45**	**8.63**	**10.25**	**11.16**	**11.37**

Table 4. Effect of the size of hidden units on PESQ

# Units	-5dB	0dB	5dB	10dB	15dB	20dB
300	1.68	2.01	2.30	2.63	3.01	3.20
500	1.71	**2.08**	2.37	2.68	**3.11**	**3.32**
800	**1.76**	**2.08**	**2.41**	**2.70**	**3.11**	3.29

6.4. Learning Rate

For the Learning rate comparisons, we see a atypical disagreement between the SSNR and PESQ results. SSNR results seem to favor 0.001 learning rate while PESQ results favor a smaller learning rate of 0.0005. It is not possible to make a strong inference here from the results. Different evaluation criteria could be used to determine the best learning rate.

Table 5. Effect of the Learning Rate on SSNR

LR	-5dB	0dB	5dB	10dB	15dB	20dB
0.0005	**3.85**	5.87	7.58	8.73	9.25	9.31
0.001	3.82	**5.94**	**7.96**	**9.48**	**10.28**	**10.51**
0.01	3.65	5.67	7.64	9.09	9.85	10.07

Table 6. Effect of the Learning Rate on PESQ

LR	-5dB	0dB	5dB	10dB	15dB	20dB
0.0005	**2.05**	**2.24**	**2.45**	**3.12**	**3.41**	**3.49**
0.001	2.02	2.21	2.39	2.59	2.75	2.94
0.01	1.90	2.14	2.39	2.50	2.70	2.78

6.5. Number of Frames

Regarding the number of frames, we see that the ideal number of frames is somewhere below 15. While for SSNR results of 11 frames are mostly better, PESQ results are tied between 11 and 15 frames. To determine an ideal frame size lower frames than 11 could be tried. In our experiments we did not go below 11 as reducing the number of frames significantly increases train time.

Table 7. Effect of the number of frames on SSNR

# Frames	-5dB	0dB	5dB	10dB	15dB	20dB
21	3.69	5.47	6.96	7.94	8.35	8.38
15	**3.90**	**5.96**	7.88	9.23	9.89	9.94
11	3.82	5.94	**7.96**	**9.48**	**10.28**	**10.51**

Table 8. Effect of the number of frames on PESQ

# Frames	-5dB	0dB	5dB	10dB	15dB	20dB
21	1.98	2.19	2.43	2.54	2.73	2.81
15	2.01	2.19	**2.40**	2.55	**2.77**	2.87
11	**2.02**	**2.21**	2.39	**2.59**	2.75	**2.94**

6.6. Size of Training Set

Comparing the trained models with different train set size, we see that the model becomes noticeably more successful as train set size is increased. This noticeable increase in performance shows that there is room for improvement by using larger train dataset size. The SSNR values seem a bigger change, while PESQ results are much closer between different train set sizes.

Table 9. Effect of the size of training set on SSNR(dB)

#Utterances	-5dB	0dB	5dB	10dB	15dB	20dB
400	3.68	5.58	7.37	8.70	9.34	9.45
800	4.23	6.40	8.51	10.08	10.86	11.01
3200	**4.78**	**7.22**	**9.57**	**11.37**	**12.36**	**12.71**

Table 10. Effect of the size of training set units on PESQ

#Utterances	-5dB	0dB	5dB	10dB	15dB	20dB
400	1.64	1.96	2.26	2.54	2.98	3.14
800	1.71	2.08	2.37	2.68	**3.11**	3.32
3200	**1.76**	**2.17**	**2.42**	**2.72**	**3.11**	**3.42**

6.7. Comparison with Traditional Methods

In this part we compare the SSNR and PESQ of the original noisy speech, the speech restored by our DAE, Log-MMSE[8], and Wavelet Shrinkage [9] (WS). We used the DAE with the best performance (# hidden units = 500, # training utterances = 3200, # stacked audoencoders = 3, # frames = 11, weights untied and symmetric structure).

Table 11. SSNR(dB) of denoised signal under different methods

Method	-5dB	0dB	5dB	10dB	15dB	20dB
Original	-7.15	-4.77	-1.83	1.57	5.34	9.37
MMSE	-1.45	-0.17	1.26	2.80	4.19	5.29
WS	-5.63	-3.44	-0.69	2.44	5.84	9.36
DAE	**4.78**	**7.22**	**9.57**	**11.37**	**12.36**	**12.71**

Table 12. PESQ of denoised signal under different methods

Method	-5dB	0dB	5dB	10dB	15dB	20dB
Original	1.53	1.79	2.08	2.38	2.75	3.08
MMSE	**1.87**	**2.15**	**2.48**	**2.77**	**3.18**	**3.45**
WS	1.41	1.75	2.08	2.40	2.82	3.15
DAE	1.76	2.17	2.42	2.72	3.11	3.42

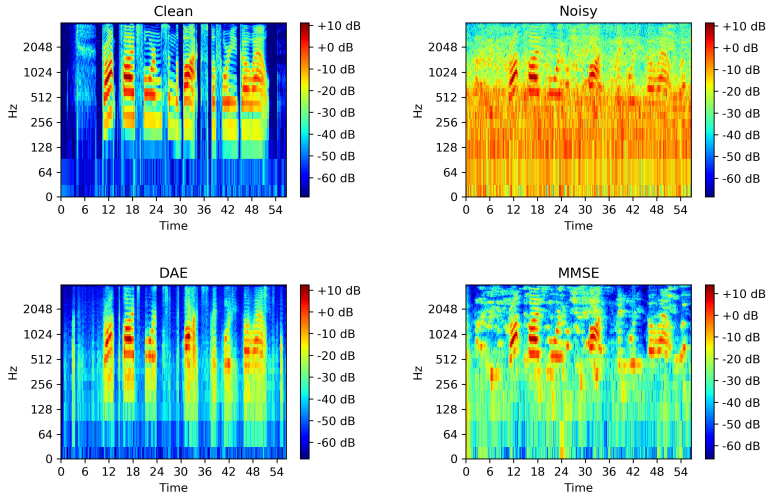

Figure 3. Clean speech power spectrogram (upper-left), and noisy speech power spectrogram (upper-right); restored speech power spectrogram based on DAE (lower-left) and restored speech power spectrogram based on MMSE (lower-right).

From the results above we see that our DAE, though having the best SSNR, does not have better PESQ than the MMSE does. On the other hand, MMSE almost achieves the worst SSNR, which means there is no direct relation between SSNR. Hence, we cannot say that our model "beat" MMSE. There are several reasons for it: (1) DAE, while removing the noise, also removes some speech components. This can be seen in the power spectrogram above (The red parts in the power spectrogram restored by DAE are less compared with the clean power spectrogram). (2) We have

limited computing resources. From the part "effect of size of training set", we see that the more the training data , the best the result. Although we have the data, we cannot fully make use of it.

6.8. Performance on Unknown Noise Types

In this part we compare the results for: Noisy Test recordings with noise types that are known by the trained model (alarm, babble, destroyer-engine and white) and noisy Test recordings with noise types that are unknown by the trained model (pink, volvo).

Table 13. SSNR of restored speech with known and unknown noise types

Method	-5dB	0dB	5dB	10dB	15dB	20dB
Known	**5.26**	**7.63**	**9.95**	**11.67**	**12.58**	**12.81**
Unknown	3.82	6.40	8.80	10.77	11.93	12.52

Table 14. PESQ of restored speech with known and unknown noise types

Method	-5dB	0dB	5dB	10dB	15dB	20dB
Known	**2.05**	**2.33**	**2.62**	**2.87**	**3.31**	**3.45**
Unknown	1.24	1.70	2.09	2.42	2.94	3.15

From the table we can see that the performance on unknown noise types are not as good as the ones on known noise types. However, it does enhance the speech quality when compared with the original noisy speech. Hence, our model, trained by 4 noise types, can actually generates to other noise types.

7. Conclusion

As shown in the "comparison with traditional methods" part, our model actually does not "beat" the MMSE method, and we have already mentioned the possible reasons there. Some other issues of our system may root in the features we extract from the data (i.e. log power spectorgram). Our DAE only makes use of the log power spectrum, but it does not make use of the phase. Phase distortion is usually difficult to be captured by SSNR, but it will be captured by PESQ. Besides, log power spectrogram cannot achieve high time and frequency resolution at the same time, which limits the performance of our model. Currently, the state of the art technique for speech enhancement is using advanced networks like Wavenet[10] and SEGAN[11]. These networks are trained directly on raw audios, therefore avoiding the disadvantages of the spectrogram. However, our system still shows a decent amount of ability to denoise the noisy speech even trained with limited data.

References

[1] Xu, Yong, et al. "An experimental study on speech enhancement based on deep neural networks." IEEE Signal processing letters 21.1 (2014): 65-68.

[2] Garofolo, John S."TIMIT acoustic phonetic continuous speech corpus." Linguistic Data Consortium, 1993 (1993).

[3] https://github.com/philipperemy/timit

[4] http://spib.linse.ufsc.br/noise.html

[5] Lu, Xugang, et al. "Speech enhancement based on deep denoising autoencoder." Interspeech. 2013.

[6] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980(2014).

[7] Rix, Antony W., et al. "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs." 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221). Vol. 2. IEEE, 2001. 1

[8] Shrawankar, Urmila, and Vilas Thakare. "Noise estimation and noise removal techniques for speech recognition in adverse environment." International Conference on IIP. Springer, Berlin, Heidelberg, 2010. 1

[9] Kim, I-Jae, Sung-Il Yang, and Y. Kwon. "Speech enhancement using adaptive wavelet shrinkage." ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570). Vol. 1. IEEE, 2001. 1

[10] Aaron van den Oord, , et al. "Wavenet: A generative model for raw audio", arXiv:1609.03499. 1

[11] Santiago Pascual, Antonio Bonafonte, Joan Serr. "SEGAN: Speech Enhancement Generative Adversarial Network", arXiv:1703.09452. 2

3

4

5

5

6

6

References