Denoising Convolutional Autoencoders for Noisy Speech Recognition

Mike Kayser
Stanford University
mkayser@stanford.edu

Victor Zhong
Stanford University
vzhong@stanford.edu

Abstract

We propose the use of a deep denoising convolutional autoencoder to mitigate problems of noise in real-world automatic speech recognition. We propose an overall pipeline for denoising, experiment with a variety of network architectures and configuration parameters, and report results on an intermediate reconstruction error metric. Our experiments show that optimal configuration for convolutional denoising varies significantly from that for standard CNN tasks such as image recognition. Further, the proposed method easily beats a simple baseline and anecdotally displays interesting behavior. A number of avenues for further research are discussed.

1. Introduction

Automatic speech recognition (ASR) is a fundamental task for a variety of real-world systems such as speech transcription and intelligent assistants. However, ASR in real, noisy environments is an ongoing challenge. For example, background noise from a cafe or from wind can significantly reduce speech recognition accuracy.

One approach to making ASR robust to noise is to *denoise* the input raw audio before processing by deconvolving or filtering out noise. Another is simply to add noisy exemplars to training. In this work, we will take the first approach and apply convolutional neural networks to audio input denoising.

2. Related Work

Audio denoising has been approached using techniques such as non-negative matrix factorization, training separate noise and voice GMM's, and noise-invariant feature extraction [6]. Neural networks have also been previously applied to this task, e.g. the recurrent neural network approach of [3].

CNN's have been applied to general speech recognition [1] and to distant speech recognition, a subgenre of noisy speech recognition [5]. Our work differs from the latter because we focus on denoising as a discrete task, and we focus on raw spectral representations rather than more processed filterbank outputs.

It is also notable that convolutional networks have been used for potentially related tasks, such as image deblurring. Since image blur and audio reverberation (one form of noise) can both be seen as convolutions, one may surmise that CNN's success in image deblurring implies potential for CNNs for at least some types of audio denoising.

3. Methodology

In this section, we specify the details of the denoising task, and introduce the denoising convolutional autoencoder that seeks to produce the spectrogram corresponding to the clean audio track given the spectrogram of the noisy audio track.

3.1. Choice of Technical Approach

We envision two major ways to apply CNN's to speech denoising, assuming the existence of a baseline speech recognition system. In the first approach, we denoise a spectrogram, and extract Mel Frequency Cepstral Coefficient (MFCC) features deterministically for use in the speech recognizer.

Alternatively, we could use a CNN to directly generate MFCC's, e.g. by adding a multilayer perceptron to the final layers of the network. These two approaches can be seen in Figure 1.

In this project, we primarily focus on method 1.

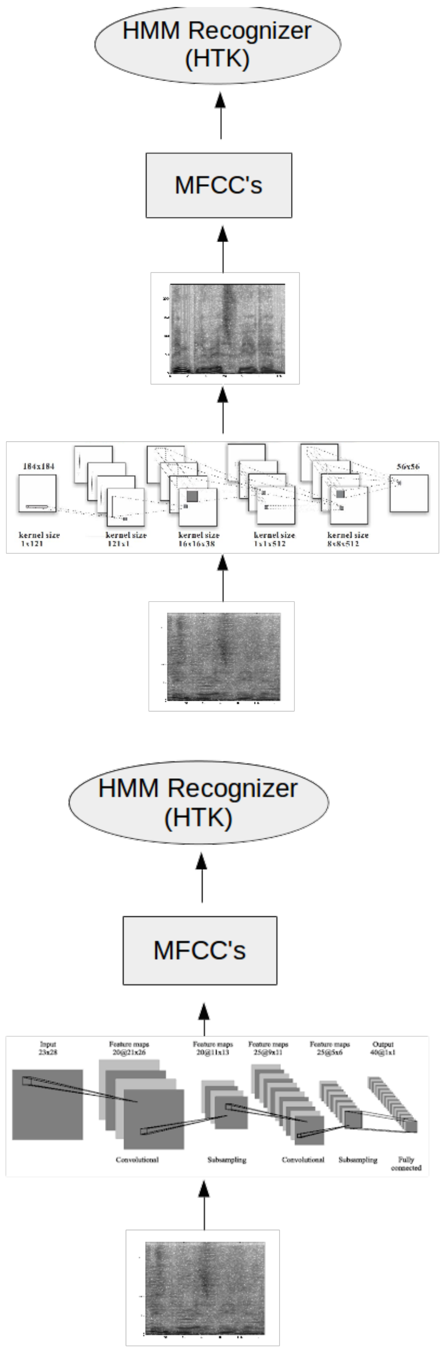

Figure 1. Two possible approaches for CNN denoising. In method 1 (top), a noisy spectrogram is given to the CNN, which produces a cleaned spectrogram. The cleaned output is used to generate MFCC's using a deterministic procedure. The MFCC's are used directly by an HMM-based speech recognition engine, such as HTK [2]. In method 2 (bottom), a multilayer perceptron appended to the CNN directly produces MFCC features, which are used as before by the speech recognizer.

3.2. Task Definition

In our work we focus on the *reconstruction* subtask, rather than the full ASR task. In particular, if C denotes a spectrogram of a clean audio snippet, and N denotes the corresponding artificially noised instance, then we attempt to learn a mapping $\mathcal{F}_\theta$, with parameters θ, which can approximately generate C given N as input. Then, given a set of training pairs $T = \{(C_i, N_i) : 1 \leq i \leq n\}$, we measure performance as mean squared error of the predicted clean output compared to the true clean output:

$$Loss(\theta, T) = \frac{1}{n} \sum_{i=1}^{n} (\mathcal{F}_\theta(N_i) - C_i)^2 \qquad (1)$$

Our focus on this metric has the downside that it does not allow us to directly compare with other denoising methods, such as other submissions to the CHiME challenge as described below. However, it afforded us maximal time during the brief duration of the project to focus on maximizing the CNN's denoising performance, rather than setting up more complex experiment pipelines requiring integration with a speech recognizer.

Note that if the spectrogram is appropriately discretized, the result is a grayscale image heat map of energies at each frequency at every time slice (see figure 2).

3.3. Dataset

The CHiME challenge [6] is a competition affiliated with a workshop at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). We use the noisy speech recognition dataset from Track 1 of the 2013 CHiME challenge to train and evaluate an audio-denoising convolutional network. The CHiME dataset consists of clean and corresponding artificially mixed noisy audio tracks. The competition organizers claim that the noise-mixing techniques used in data preparation are highly representative of typical noise encountered in a real application. The training data consists of 17,000 very short utterances that make use of a closed-class and extremely limited vocabulary. The development and test data consist of 600 similarly structured utterances. Each test utterance is mixed with various levels of background noise, so that one can measure the average performance of a system at various noise levels.

3.4. Denoising Convolutional Autoencoder

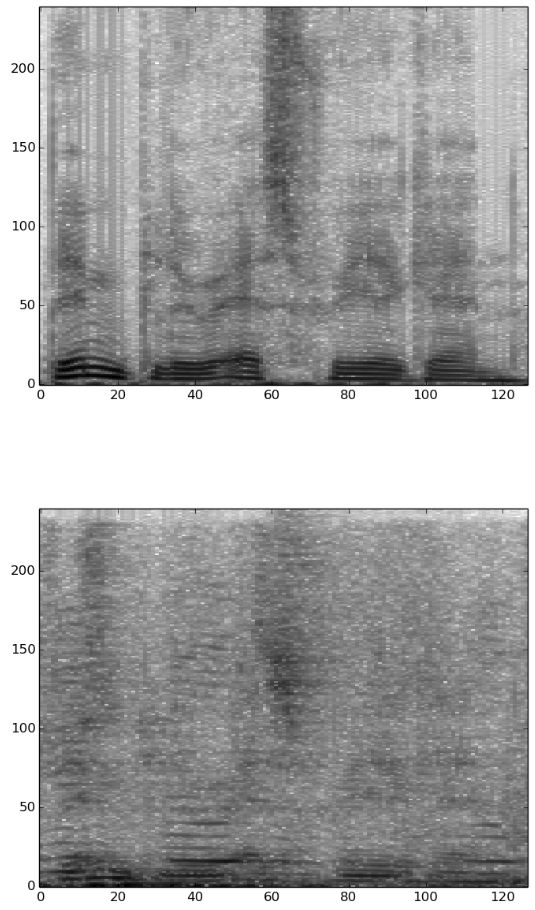

Figure 2. spectrograms of the clean audio track (top) and the corresponding noisy audio track (bottom)

There is an important configuration difference between the autoencoders we explore and typical CNN's as used e.g. in image recognition. In particular, our CNN's do not use any pooling layers, as the purpose is to reconstruct an output of the same dimensions as the input. Pooling followed by up-sampling is also a possibility, which we have not explored.

4. Experiment

4.1. Preprocessing

We train denoising convolutional autoencoders with varying architectures using 100 audio tracks from the CHiME challenge. Due to the varying lengths of the audio tracks, we further randomly sample fixed-sized windows from each audio track. Namely, for each clean-noisy audio track pair, we perform the following:

1. Apply fast Fourier transform on the audio tracks to obtain corresponding spectrograms. For our experiments, we apply FFT on 10 millisecond time windows.

2. Randomly subsample corresponding fix-sized windows from each spectrogram pair. For our experiment, we collect windows that are 100 time units and cover the entire frequency range. This corresponds to 1 second of audio.

3. Because the spectrograms have a large dynamic range (eg. from 10^{-3} to more than 10^{10}), we apply element-wise log-transform.

4. Finally, we apply zero-mean unit-variance normalization on the spectrograms.

In our experiments, the log transformation and data normalization proved essential for convergence, due to the nature of the regression loss and the extremely large dynamic range of the input spectrograms.

4.2. Training

The result of the above preprocessing is taken to be the input and output of the learning task. Namely, the normalized spectrogram corresponding to a window in the noisy audio track is given as input to the convolutional autoencoder, and the normalized spectrogram corresponding to the same window in the clean audio track is given as target to the convolutional autoencoder.

In all cases, we train the models with stochastic gradient descent and Nesterov momentum, which provided minor improvements over Adagrad. We use L2 regularization with an extremely small weight of 0.00001. In general, underfitting is a far greater challenge for our task than overfitting.

We ran our experiments using the GPU image provided by Terminal.com and a NVIDIA GTX760. In both cases, we implement our models and experiments using Caffe [4].

5. Results

Example outputs of the autoencoder are compared to spectrograms of the noisy and clean audio tracks in Figure 3. It is unclear what to conclude from these examples. One possibility is that the network results in a sort of Gaussian blur over the noisy input. Another is that something more sophisticated is happening, as it does appear that meaningful structure from the clean data is being preserved.

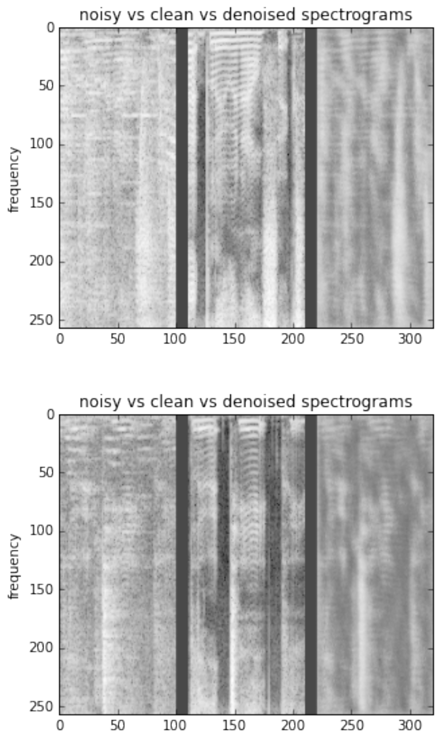

Figure 3. Denoised spectrograms from the autoencoder. Each image is divided into three vertical segments. The left-most segment is the spectrogram of the noisy audio track. The middle segment is the spectrogram of the clean audio track. The right-most segment is the output of the autoencoder.

5.1. Architectures vs. Activation

We compare against an extremely simple and easy to compute baseline, namely a single-layer, 1x1 affine convolutional network with no nonlinearities. The intuition is that this can correct simple problems of scaling in the noisy input, owing to the different means and variances between the noisy and clean data. The baseline mean squared reconstruction error is 2.55. Due to time and resource constraints, we have not yet computed end-to-end speech recognition experiments, which would permit a variety of comparisons with stronger baselines using speech recognition accuracy metrics.

The results across various architectures are shown in Table 1 and Table 2. In order to preserve information throughout the network, we do not use max-pooling. Empirically, we found that the parameters required for fully-connected layers at the end of the network are better utilised implementing additional convolutional layers. This matches our expectation, since denoising is essentially a *local* operation which is informed by nearby regions of the spectrogram.

Amongst the architectures we explored are cascading long filters, which were found to yield performance improvements in image deblurring in [7]. For our experiments, the intuition is that convolutional filters long in the time domain may be able to capture patterns across time that may help in identifying noise. However, in practice, we did not find noticeable performance gains resulting from long convolutional filters.

Another architecture we explored was adding network-in-network. This gave similar performance to using standard large filters, and did not result in noticeable performance gains.

In Table 1, we note that Tanh activation consistently outperformed rectified linear units across all architectures. Our hypothesis of why this may be happening is that the severity of the regression loss may push rectified linear units to the zero-gradient region, and the autoencoder cannot recover from the dead ReLU units. Whereas in classification tasks, activation sparsity may be desirable, in autoencoding, it may cause undesirable information loss. For reference, the variance in the clean spectrogram is 2.53, and the average mean squared error between the noisy spectrogram and the clean spectrogram after mean normalization is 2.40.

5.2. Depth vs. Filter Size

We also experimented with varying filter sizes across layers of varying activation depth. In the event that we use a shallow activation depth, we construct a deeper network to leverage a similar number of parameters.

In Table 2, we show that larger filter sizes consistently outperform smaller filter sizes for the autoen-

CNN Architecture	Description	ReLU	Tanh
7x1x16 1x7x16 7x1x16 1x7x16 7x7x16 7x7x16 7x7x1	long skinny	2.10	1.90
1x7x16 7x1x16 1x7x16 7x1x16 7x7x16 7x7x16 7x7x1	long skinny	2.10	1.82
3x3x32 3x3x32 3x3x32 3x3x32 3x3x1	cascaded 3x3	2.02	1.80
7x7x16 7x7x16 7x7x16 7x7x8 7x7x16 7x7x16 7x7x16 7x7x1	8-layer 7x7	1.96	1.78
7x7x16 1x1x32 1x1x32 1x1x32 7x7x16 7x7x16 7x7x1	network-in-network	1.82	1.76
7x7x16 7x7x16 7x7x16 7x7x16 7x7x1	5-layer 7x7	1.82	**1.74**

Table 1. Mean squared reconstruction error of held-out development data, across varying architectures. The baseline reconstruction error is 2.55.

Activation depth	3x3	5x5	7x7
128-1	2.10	1.96	1.88
48-48-1	2.00	1.84	1.74
24-24-24-24-1	1.90	1.76	**1.74**
16-16-16-16-16-16-1	1.98	1.88	1.74

Table 2. Mean squared reconstruction error across varying depth of activation volume. For example, "128-1" means there is one layer of the given filter size with 128 output channels, and one layer with 1 output channel. In all cases, the final layer has no nonlinearity, and all nonfinal layers have Tanh nonlinearity. The baseline reconstruction error is 2.55.

coding task, regardless of the depth of the network. One reason why this may happen is that because we do not use max-pooling, larger filter sizes offer a means by which the network can collect observations from a larger area, which is crucial for identifying patterns in the spectrogram.

A second notable observation is that very deep networks seem not to be required to get the best reconstruction performance.

Given the reconstructions we have achieved, some which are shown in Figure 3, and numerical reconstruction errors shown in Table 1 and 2, we believe convolutional autoencoders deserve further investigation.

6. Future Work

One direct continuation of this work is to present the reconstruction of the autoencoder to a down-stream speech recognizer such as HTK [2]. Given our empirical evidence that the reconstruction is closer to the spectrogram of the clean audio track, we hypothesize that a downstream speech recognizer such as HTK may produce more accurate outputs (eg. in terms of word error rate). In order to use the reconstructions in HTK, we can extract MFCC features from the reconstruction through a deterministic procedure. The MFCC features would then be used as input into HTK. Alternatively, we could employ method 2 of Figure 1, and use a network to generate MFCC's directly.

7. Conclusion

In this project, we investigated the use of convolutional autoencoders for audio denoising. In our experiments, we found that large filter sizes and Tanh activations consistently achieved higher performance than small filter sizes and linear rectifiers across varying architectures. Finally, we presented empirical evidence in the form of qualitative comparisons as well as quantitative reconstruction error between the output of the autoencoder and spectrograms of clean audio tracks to demonstrate that convolutional autoencoders are a potentially promising approach for speech denoising.

References

[1] O. Abdel-Hamid, A. rahman Mohamed, H. Jiang, and G. Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing 2012*.

[2] Cambridge University. Htk speech recognition toolkit.

[3] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns, 2014.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[5] S. Renals and P. Swietojanski. Neural networks for distant speech recognition. *2014 4th Joint Workshop*

on Hands-free Speech Communication and Microphone Arrays (HSCMA), IEEE 2014, 2014.

[6] E. Vincent, J. Barker, S. Watanabe, J. L. Roux, and F. Nesta. The second CHiME speech separation and recognition challenge: An overview of challenge systems and outcomes. *IEEE International Conference on Acoustics, Speech and Signal Processing, 2013*.

[7] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. *Advances in Neural Information Processing Systems, 2014*, 2014.