Final Project

Matthew Markowitz, Lifu Xiao

December 1, 2018

1 Introduction

The database is a set of noisy recordings[1], which have poor quality for further usage. So it make sense to improve them. In order to remove the noise, we propose to use online dictionary learning[2].

There were many challenges involved with this problem. One problem involved the large number of samples that were needed to begin the dictionary updates. Unfortunately, we can't update the dictionary until every atom has been used at least once. To overcome this, we refused to update the dictionary until all the zero elements were filled. However, this is also not ideal because this requires substantially more training data as the window size for our audio increased. Experimentally, this can be overcome by adding a random small constant before we start (such as 0.000000001), however, this is little to no theoretical backing for this, so we did not take this route.

For the problem that we are trying to solve, the state of the art involves using a technique called Spectral Subtraction[3]. The general idea in this technique involves estimating the spectrum of background noise from pauses in speech. however there are some limitations. It assume that the background noise in uncorrelated with the speech and there are pauses in the audio that can be used to estimate the background noise. This may not be true for all audio signals. Also, it assumes that the spectrum of background noise is constant which it may not be.

Convolutional neural networks also performance well in audio denoising. An end-to-end learning method based on wavenet[4] that learns in a supervised fashion via minimizing a regression loss and it reduces the time-complexity. Another widely used method is deep recurrent neural networks[5] which have good performance on extracting acoustic features from noisy data. To be more specific, Long short-term memory networks are used to avoid the vanishing gradient problem. It use a parallel database of noisy and clean acoustics parameters as input and output. It produce a good result but it have a high computation cost. So we try to apply

2 Problem Statement

We used a python library known as librosa[6] to import our audio data. The audio recordings found in our test set had a sampling rate of 22050. This meant that every second of audio held approximately 22,000 numbers to represent it. For this reason, down sampling became necessary. Although some sacrifice in audio quality was necessary, we were able to reduce the sampling rate to 5,000, which made our calculations more feasible. The 5,000 points per second was still computationally intensive, but we found that we could break each second into X millisecond windows to ease computation further without sacrificing much quality.

3 Algorithm

3.1 Data Preparation

Initializing the $\boldsymbol{A}_0 \in \mathbb{R}^{k\times k}$ and $\boldsymbol{B}_0 \in \mathbb{R}^{m\times k}$ as $\vec{\mathbf{0}}$
k is atoms number and m is the dictionary size.

3.2 Sparse Coding

When each x_t come, using LARS[7] to calculate

$$\alpha_t \triangleq \underset{\alpha\in R^k}{\arg\min} \frac{1}{2}\|\boldsymbol{x}_t - \boldsymbol{D}_{t-1}\alpha\|_2^2 + \lambda\|\alpha\|_1$$

where $\boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{D} \in \mathbb{R}^{m\times k}$ and $t \leq T$(maximum number of iterations)
Then updating $\boldsymbol{A}, \boldsymbol{B}$ with mini-batch by

$$\boldsymbol{A}_t \leftarrow \boldsymbol{A}_{t-1} + \frac{1}{\eta}\sum_{i=1}^{\eta} \alpha_{t,i}\alpha_{t,i}^T$$

$$\boldsymbol{B}_t \leftarrow \boldsymbol{B}_{t-1} + \frac{1}{\eta}\sum_{i=1}^{\eta} \boldsymbol{x}_{t,i}\alpha_{t,i}^T$$

3.3 Dictionary Update

$$\boldsymbol{D}_t \triangleq \underset{\boldsymbol{D}\in C}{\arg\min} \frac{1}{t}\sum \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{D}\alpha_i\|_2^2 + \lambda\|\alpha\|_1$$

Where $C \triangleq \boldsymbol{D} \in \mathbb{R}^{m\times k}$ s.t. $\forall j = 1, ..., k, \boldsymbol{d}_j^T\boldsymbol{d}_j \leq 1$ to ensure the convex.
Using block-coordinate descent to update dictionary Extracting columns of $\boldsymbol{A}$ and $\boldsymbol{B}$

$$\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_k] \in \mathbb{R}^{k\times k}$$

$$\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_k] \in \mathbb{R}^{k\times k}$$

for each column from $j = 1 \Rightarrow k$

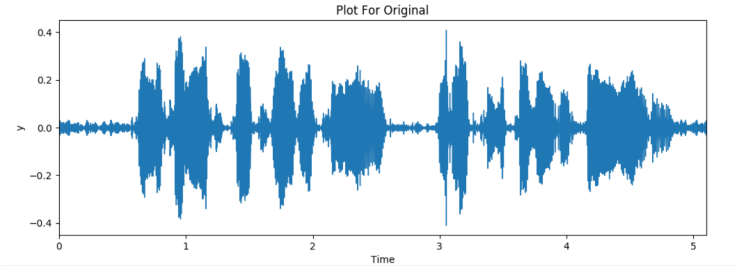
$$\boldsymbol{u_j} \leftarrow \frac{1}{A[j,j]}(\boldsymbol{b}_j - \boldsymbol{D}\boldsymbol{a}_j) + \boldsymbol{d}_j$$

$$\boldsymbol{d}_j \leftarrow \frac{1}{\max(\|\boldsymbol{u}_j\|_2, 1)}\boldsymbol{u}_j$$

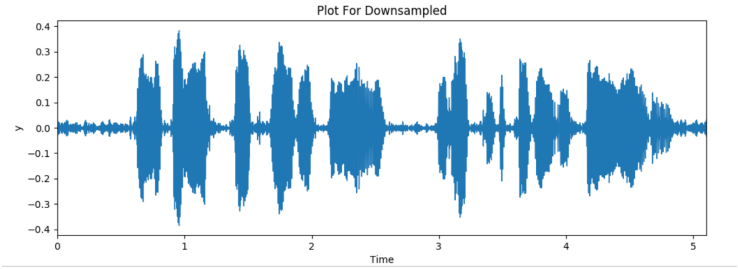
return $\boldsymbol{D}$ for next iteration

4 Experiments

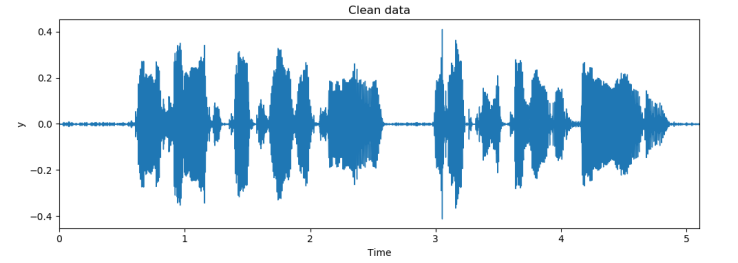
To visualize our result, we picked a 6 seconds segment of the output audio. The Original, Downsampled figures and the Clean data which is used for a baseline are as follows.



(a) Original



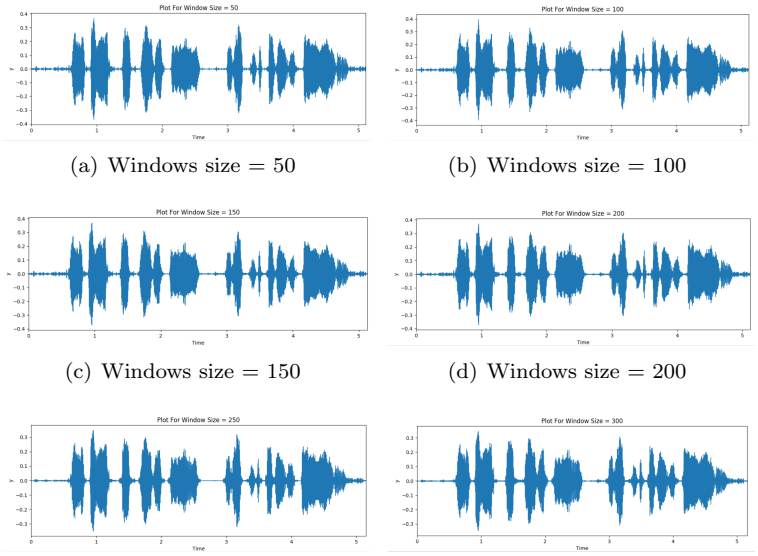
(b) Downsampled



(c) CleanData

Figure 1: Original , Downsampled and the Clean data

We set $\lambda = 0.025$ and test different window sizes for the denoising program. The results are presented in Figure 2



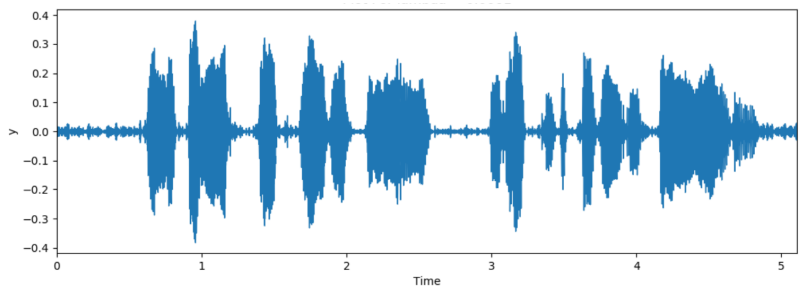
(a) Windows size = 50 (b) Windows size = 100

(c) Windows size = 150 (d) Windows size = 200

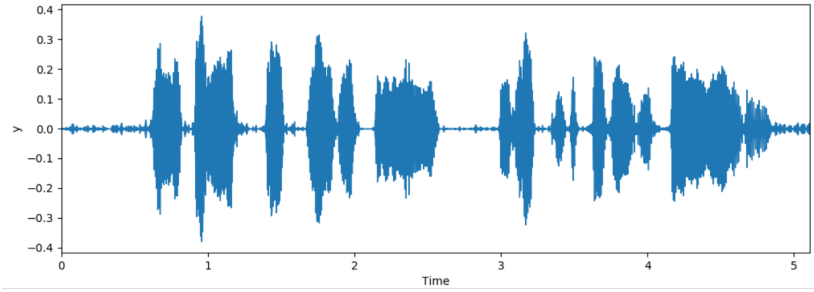
(e) Windows size = 250 (f) Windows size = 300

Figure 2: Denoising result under different window sizes

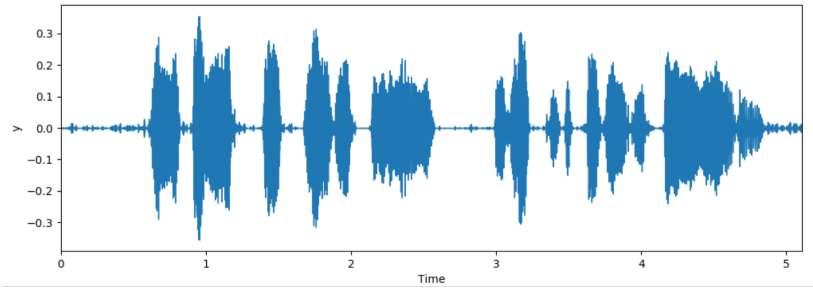
The smaller window sizes give the better result. Now we choose $k = 500$ and $m = 50$ for the online dictionary learning step and the result generated by different λ are presented in Figure 3.



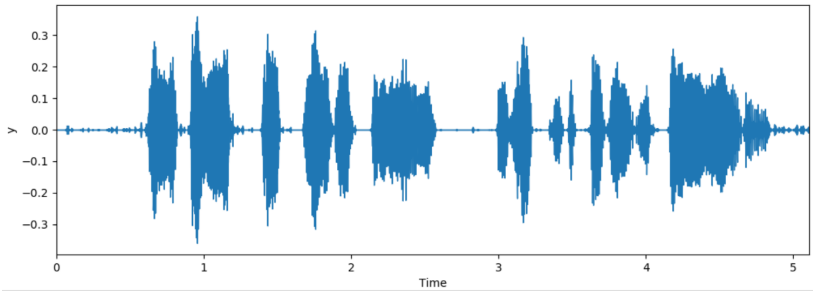
(a) $\lambda = 0.005$



(b) $\lambda = 0.025$



(c) $\lambda = 0.035$

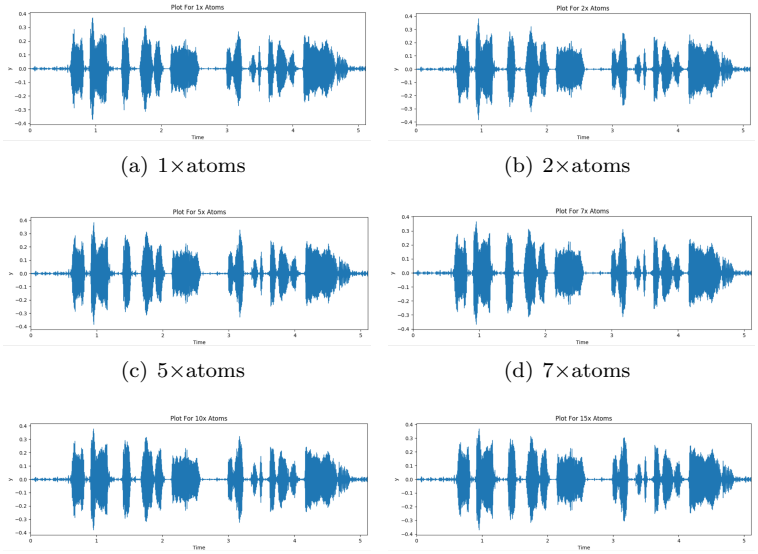


(d) $\lambda = 0.05$

Figure 3: Denoising result under different λ

We note that $\lambda = 0.025$ have the best performance among three other λs.

So we set $\lambda = 0.025$ to test different atoms. We set atoms from $1\times$ dictionary size to $15\times$ dictionary size.



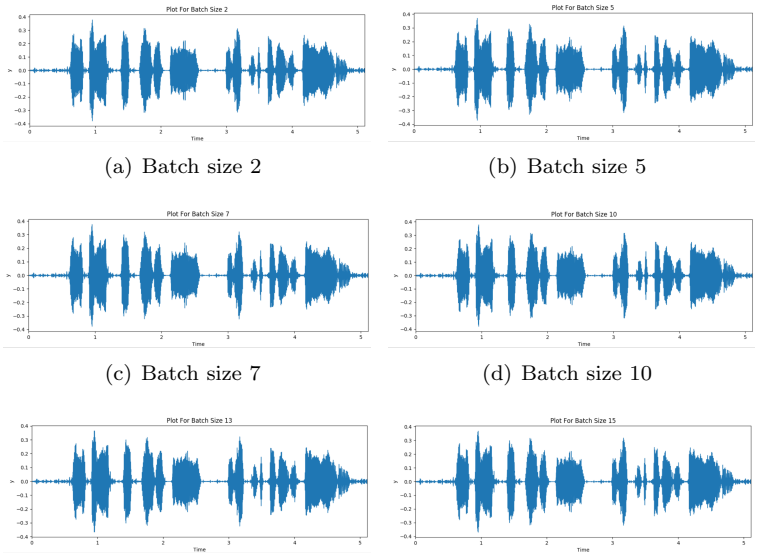
(a) $1\times$atoms (b) $2\times$atoms

(c) $5\times$atoms (d) $7\times$atoms

(e) $10\times$atoms (f) $15\times$atoms

Figure 4: Denoising result under different atoms

After we examine the output, the $15\times$atoms turned to have a better precise. We also test different batch sizes which are illustrated as follows.



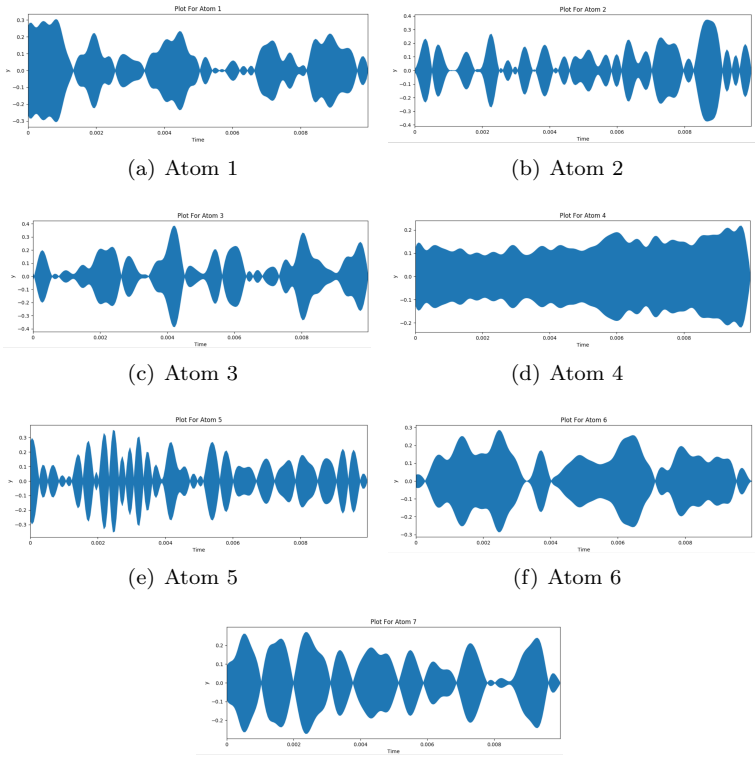
(a) Batch size 2 (b) Batch size 5

(c) Batch size 7 (d) Batch size 10

(e) Batch size 13 (f) Batch size 15

Figure 5: Denoising result under different atoms

Finally we randomly pick seven atoms and visualize them to show that how do they work.



(a) Atom 1

(b) Atom 2

(c) Atom 3

(d) Atom 4

(e) Atom 5

(f) Atom 6

(g) Atom 7

Figure 6: Atoms

Our experiments have showed that online dictionary learning is very efficient and robust on audio denoising problem. Most of our judgements are based on the output audio files which is not ocular. An audio waveform provides a more direct way to make comparison, however, the difference is not easy to be noticed. Finding another judgement model can be very interesting.

References

[1] Cassia Valentini-Botinhao et al. Noisy speech database for training speech enhancement algorithms and tts models. 2017.

[2] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[3] Navneet Upadhyay and Abhijit Karmakar. Speech enhancement using spectral subtraction-type algorithms: A comparison and simulation study. *Procedia Computer Science*, 54:574–584, 2015.

[4] Dario Rethage, Jordi Pons, and Xavier Serra. A wavenet for speech denoising. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5069–5073. IEEE, 2018.

[5] Cassia Valentini-Botinhao, Xin Wang, Shinji Takaki, and Junichi Yamagishi. Speech enhancement for a noise-robust text-to-speech synthesis system using deep recurrent neural networks. In *Interspeech*, pages 352–356, 2016.

[6] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.