CRITIQUE OF PRESENTATION BY GROUP 19: *Voice Denoising with Deep Learning*

This project uses a U-Net like model architecture to denoise samples of human voice audio.

Critiques by **Group51**

- Very cool work to construct your own dataset in this scenario
- Very detailed theoretical explanation of the neural network architecture
- Great work in also producing the physical example of denoised output
- Organized workflow in the code notebooks; having a demo is also commendable
- Have you considered the impact of settings on various spectrogram methods?
- Can you report results to support the advantages of pruning for this project?
- Could you justify why you used a smaller validation set? Have you taken all measures to reduce overfitting? If yes, what moves did you make to do so?

Response:

Thanks for the praise and love! We respect it!

For the first question, the answer is yes. I mean we do have considered the impact of settings on various spectrogram methods but there are too many things need to be debugged. Like the frame length, the hop length. Finally we just fixed these parameters.

For the next question, the aim of pruning is to accelerate the training process. So it's hard to show a result for that part. I can say that before the pruning, we will need like 1min-1.5min for one epoch. After pruning we only need 30s for one epoch. Also we didn't prune all the layers. For the conv2D layers, we only pruned the last several layers.

For the last question, I'm not sure what does 'a smaller validation set' mean. The training set always larger than the validation set, isn't it? Although overfitting is not a big problem in our project, I still tried some methods to solve it. Actually, adding dropout layers really help. I started from adding one dropout layer after each conv2D layer. Then tried to reduce the number of dropout layers and reduce the size of it. Also I tried to enlarge the batch size.

Critiques by **group 67.** Group 19 did a good job in separation of the background noise and the voice data from human. The model was explained well and special techniques used are elaborated. While the following are some recommendations related to presentation:

● During the presentation, most of the contents are good, but one of the member's voice was too low to hear.

● It would be better if we know what kind of the loss function you used for both encoder and decoder. Because the model loss fluctuation in the test data could be very interesting to look into.

● The audio demonstration is very good. One confusion that was perceived is that what is essential difference between selecting the right neurons and applying dropout and the pruning method mentioned? Your team could try to mention this in the report.

● The STFT decomposes the dataset into the spectrogram and the phase information. On the spectrogram we see that the noise part is directly subtracted from the original data. This might cause a loss in the voice if some information is in that band, you can try attenuation.

● Also, it could be interesting if you apply some network in training the phase information, people use phase retrieval techniques to better separate out the useful information from sound signals. Since you are using conv2d, I think that the phase information could also be analyzed very well.

● It will also be interesting if you mix the male and female like voices together, or even sound from different people (for a dataset) while training, this would generate more robust and unbiased models.

Response:

Thank you for your valuable comments. I will explain your doubts one by one as follows.

- Because this is a difficult time for all of us now, our team recorded this video using zoom from different places. We apologize for the inconvenience.
- In fact, U-Net is not pure encoder-decoder networks. Unlike the conventional encoder/decoder structure, these two parts are not decoupled. Skip connections are used to transfer fine-grained information from the lower layers of the analysis path to the higher layers of the synthesis path, because this information is needed to generate reconstructions with accurate fine-grained details.
- Thanks for the advice. Selecting the right neurons can help us to get better results while training our model, which means to make our model achieve the goal better. Then, applying dropout can help to avoid the overfit problem. So, we can make our model perform better in a more general case. Besides, pruning can help to eliminate useless connections between neurons. After pruning, the model can use fewer resources and spend less time doing almost the same work, which usually can speed up the inference while using fewer resources. I hope this can make these three different methods clearer.
- Since the goal of training our model is making it detect the noise from the input spectrograms. As we can see from the result, our model performs very well, which means it can almost perfectly detect the noise from the noisy sound. Using attenuation is also a good idea, we will try to add it and do some comparison in the future.
- As we mentioned, from the intuitive look, we can tell that a lot of valued information is in the spectrogram. For the phase, it looks very chaotic. But this may not eliminate the possibility of the method you mentioned. However, because we only have limited time to work on this project, we chose a more reasonable way to do our project. So we chose to use spectrograms.
- We apologize for not making the dataset clear. In the dataset of the human speaking voice, we used in this project, the voice is from different speakers including both male and female.

Critiques by **group 56.**

- This group explored denoising audio by applying a “UNet” to a spectrogram to predict the noise spectrogram only. The UNet architecture describes a two-dimensional convolutional autoencoder with feedforward connections, or “skip connections.” All spectrograms were constructed using the short term Fourier transform (STFT) and magnitude was passed through the network only. The training set was constructed with samples from two databases: LibriSpeech for voice, ESC-50 for environmental sound; the resulting “noised” speech was fed in as training input, and the training label used was the noise content only. After training, the predicted noise was then subtracted from the noised speech to recover a clean speech spectrogram. An inverse STFT was then applied to the recovered spectrogram and the phase (left untouched at STFT) to yield the final deoised signal. Dropout layers and batch size were explored to avoid converging at local minima of the objective function, and pruning was applied to the network by dropping neurons with the least-active weights, calculated using a mean of L1 and L2 norms. The pruned network was then trained further to compensate for the change. This was a nice holistic presentation that focused on not only network construction and training, but also iteratively changing the network architecture (pruning).
- Training this well-known architecture with pruning seems to offer a way to build a robust network, which could be heard when you played back the audio samples (good idea to incorporate that in the presentation). Because speech noise reduction seems to be one of the more popular applications of machine learning, it gets overwhelming looking at all of the different ML-based approaches. I’d love to see some more explicit discussion on how this specific network and pruning method compares to other well-known networks.
- What loss function was being used?
- Does pruning apply to entire neuron layers and their L1/L2 mean? Slide 13 mentions that removing too many neurons in a pruning cycle may damage a network beyond repair. What does that behavior look like in terms of training history (loss vs. epoch), especially in contrast to training batchsize/dropout (slide 11)? Does it affect local extrema more or less, or is it a different change altogether?
- I liked the mention of changing STFT parameters as a point of future research. I would love to see how DFT size, window type, and overlap affect spectrograms (some straight signal processing work), and how that in turn affects the model being trained over the spectrograms.

Response:

- Thanks for your valuable suggestions. Actually the pruning is always the core part of our model, we spent lots of time on designing the structure.
- Actually I myself have previously tried human-voice separation with LSTM and RNN, the effect is quite poor, as much human voice were removed and noise being preserved. Unet is actually performing very well.
- The loss function we used is Huber loss, since it is less sensitive to outliers in data than the squared error loss. In our scenario, it is better than just using the MSE.
- The pruning is only for the last 2 layers of convolutional layer in encoders and decoders, not all layers were pruned.
- The number of STFT would be our future research target. However, since that only affects the quality of the sound after it turns back, we would expect to see the more number we use, the better quality the voice would have.

Voice Denoising with Deep Learning(Group 19)

Haoyang Ding A53320920 | Yu Wang A53312506 | John Xie A53307207 | Shang Wang A53320852

Abstract

Speech audio sometimes has environmental noise, which makes it harder for people to catch the key idea. We want to construct an audio denoise system to get clean speech audio. Image noise reduction is a common problem and we can convert the audio denoise problem into image noise reduction problem by using the amplitude spectrum representation of the sound. In this project, we use neural networks to perform noise reduction on audio with speech to eliminate the noise and make the speech clearer.

1. Introduction

Nowadays, there are microphones that can filter the background noise, but when listening to video or audio containing speech voice, there will be noise that interferes with the acquisition of voice content in background. There are many kinds of noise in the video or audio, for example: wind, the sound of vacuum cleaners, footsteps, etc.

In today, we already have some professional software that can reduce the noise of sounds, but most of the software needs people do the denoise process manually. Take Adobe Audition as an example, it provides functions enable professionals to perform detailed sound processing but its parameters are so numerous that this kind of software is difficult for people not professional.

In this project, we analyzed the characteristics of different sounds and noises, and used neural networks ,especially U-Net, to build a audio denoise system to automatically reduce background noise, so that it can be used for most non-professionals. In Section 2, we will explain the overview of the project, from datasets to audio representations and neural networks used in the project.

2. Related Work

In this work of E.M. Grais and M.D. Plumbley.[5], they propose to use deep fully convolutional denoising autoencoders (CDAEs) for monaural audio source separation. They use as many CDAEs as the number of sources to be separated from the mixed signal. Each CDAE is trained to separate one source and treats the other sources as background noise. The main idea is to allow each CDAE to learn suitable spectral-temporal filters and features to its corresponding source.

In the work of Andreas Jansson[3], they use U-Net and audio-to-image conversion to decompose music audio signals into vocals and background sounds.

In the work of X. Zhang and J. Wu.[8], they proposed a denoising-deep-neural-network (DDNN) based VAD to address the denoising problem. In the work of K. Han[2], they proposed to perform speech dereverberation using supervised learning, and the supervised approach is then extended to address both dereverberation and denoising.In the work of D. Rethage, J. Pons, and X. Serra[6], they proposed an end-to-end learning method for speech denoising based on Wavenet.

In our project, we choose to do more after Andreas Jansson's work, we will try to use Short-time Fourier transform as audio-to-image conversion and use U-net to do the model for denoise the speech audio.

3. Datasets

To get speech audio with specific environmental noise, first we blend noises with clean voices from two different datasets. For clean English speech voice, we get data from LibriSpeech. This dataset includes about 1000 hours of 16kHz read English speech, prepared by Vassil Panayotov. The data is derived from read audiobooks from the LibriVox project, and has been carefully segmented and aligned. Also, from ESC-50 dataset, we get environmental noise[4], which includes 2000 environmental audio suitable for benchmarking methods of environmental sound classification.

4. Methods

4.1. Short-time Fourier transform

We use Short-time Fourier transformSTFT to convert the audio signal into a spectrogram and phase picture.

From the original time series to time-frequency decomposition, audio has many different representations. And it is important for us to choose a well representation to reach

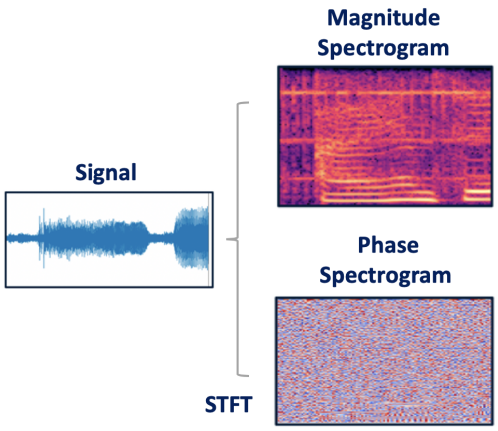

Figure 1. Process original signal with STFT.

a good performance in our system. Spectrograms have proved to be on of the most useful representations for audio processing among all[1].It consists of 2D images which represents sequences of Short Time Fourier Transform (STFT) with time and frequency as axes, and its brightness representing the strength of a frequency component at each time frame. So that they can appear a natural domain to apply the CNN-like architectures for images directly to sound. Usually, magnitude spectrograms have most of the information for the signal. Phase spectrograms show only little information compared to magnitude spectrograms. (Figure 1) In our project, we will use magnitude spectrograms to predict the noise model.

The short-time Fourier transform and the time-varying frequency response are introduced as representations for signals and linear time-varying systems[5]. Short-time Fourier transform (STFT) is a universal tool for signal processing. The Short-time Fourier transform (STFT), is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time. In practice, the procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the Fourier transform separately on each shorter segment. This reveals the Fourier spectrum on each shorter segment.

The definition of the short-time Fourier transform is as follows:

$$\sum_{n=-\infty}^{\infty} x(n)w(n-mR)e^{-j\omega n} \tag{1}$$

where $x(n)$ = input signal at time n $w(n)$ = length M window function $X_m(\omega)$ = DTFT of windowed data centered about time mR R = hop size, in samples, between successive DTFTs.

DTFT (Decrete Time Fourier Transform) is a discrete time Fourier transform. The definition of it is as follows:

$$X(\tilde{\omega}) \triangleq \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \tag{2}$$

Where x (n) is the signal amplitude at the number of samples n.$\tilde{\omega} \triangleq \omega T \in [-\pi, \pi)$

In implementation, the short-time Fourier transform is calculated as a series of fast Fourier transforms (FFTs) of windowed data frames, where the window "slides" or "hops" over time.

4.2. U-Net

In this project, we will use U-Net, which is a Deep Convolutional Autoencoder with symmetric skip connections to the clean environmental noise inside speech audio.

U-Net is a convolutional network introduced in biomedical imaging, which can improve the accuracy and location of neuron structure microscopic images[3].

It is generally agreed that a successful training of deep networks requires thousands of annotated training samples. However, U-Net relies on the use of data enhancements to use samples with more efficiency.

Our network architecture is shown in figure 2[7]. And there is a contraction path in its left side and an expansion path in its right side.The contraction path has the typical architecture of convolutional networks.It includes two same 3x3 convolutions, each follows a ReLU and a 2x2 max pooling operation with stride 2 for downsampling, which doubles the number of feature channels.Each step in the expansion path includes upsampling and then 2x2 convolution to halve the number of channels and perform the corresponding feature map cut from the contraction path Concatenation, and then two 3x3 convolutions, each followed by a ReLU.And since boundary pixels are lost in each convolution, cropping is necessary.In the last layer, 1x1 convolution is used to map each 64-component feature vector to the number of classes. The network has a total of 23 convolutional layers.

The energy function is given by a pixel-wise soft-max over the final feature map combined with the cross entropy loss function. The pixel-wise soft-max is defined as $p_k(\mathbf{x}) = \exp\left(a_k(\mathbf{x})\right) / \left(\sum_{k'=1}^{K} \exp\left(a_{k'}(\mathbf{x})\right)\right)$ where $a_k(\mathbf{x})$ denotes the activation function in feature channel k at the pixel position $\mathbf{x} \in \Omega$ with $\Omega \subset \mathbb{Z}^2$.$K$ is the number of classes and $p_k(\mathbf{x})$ is the approximated maximum-function. I.e. $p_k(\mathbf{x}) \approx 1$ for the k that has the maximum activation $a_k(\mathbf{x})$ and $p_k(\mathbf{x}) \approx 0$ for all other k. The cross entropy loss function then penalizes at each position the deviation of $p_{\ell(\mathbf{x})}(\mathbf{x})$ from 1 using

$$E = \sum_{\mathbf{x}\in\Omega} w(\mathbf{x}) \log\left(p_{\ell(\mathbf{x})}(\mathbf{x})\right) \tag{3}$$

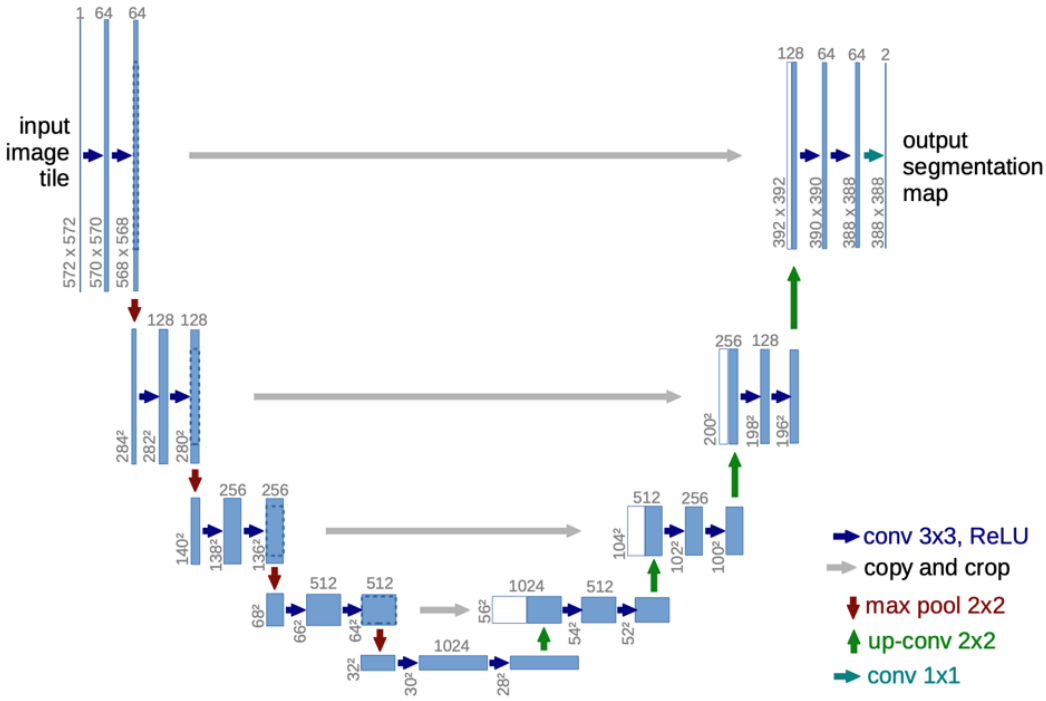

Figure 2. Structure of U-Net

where $\ell : \Omega \to \{1, \ldots, K\}$ is the true label of each pixel and $w : \Omega \to \mathbb{R}$ is a weight map.

The separation border is given by using morphological operations. And then the weight map is given as

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right) \quad (4)$$

where $w_c : \Omega \to \mathbb{R}$ is the weight map to balance the class frequencies.

5. Experiments

In this section, we illustrate the training details in our U-net model with some specific parameters and the detailed process of our prediction part. Also, some of the results are presented at the end of this section.

5.1. Training

The configuration of the encoder includes 10 convolutional layers (with LeakyReLU, maxpooling and dropout).The decoder is a symmetric extension path with skip connections. The last active layer is the hyperbolic tangent (tanh) with an output between -1 and 1. To start training from scratch, using the He Normal Initializer to set the initial weight. The model is trained using the Adam optimizer, and the loss function we choose Huber loss, as a compromise between L1 and L2 losses. Before training the model, we set the epoch to 200 and the batch size to 10.

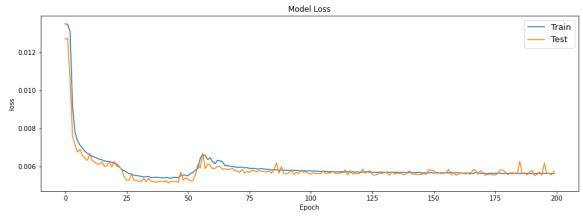

Figure 3. Huber loss of training and validation process

And to speed up the following prediction speed we also use pruning in the last two Conv2D layer in the encoder and decoder process, in another words, remove the least important neurons with a pruning rate of 10 percentage, we successfully accelerate the speed of prediction.

At the end, we obtain the training loss of 0.0056 and the validation loss of 0.0058. The Huber loss during the training process is shown in Figure 3.

5.2. Prediction

During the prediction process, the noisy speech audio is converted into a Numpy time series for 1 second. Each time a series of signals are converted into a magnitude spectrogram and a phase spectrogram by STFT. The magnitude

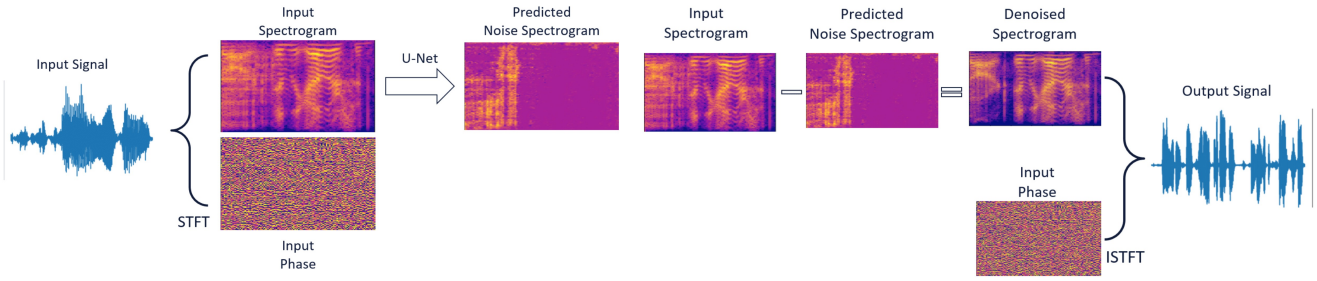

Figure 4. Signal noise reduction process.

spectrogram is used as the input of our U-Net network, which predicts the noise model for each time window.

Then subtract model from noisy speech spectrum. Using the denoised amplitude spectrum and the initial phase as the input to the Inverse Short-Time Fourier Transform (ISTFT) will get the the denoised time series and then can be converted to audio. This whole process is shown in Figure 4.

5.3. Results

In this section, we show the results of an examples against a speech voice with a mixed background noise, such as, dog's bark, alarms, door knocks, insects and so on.

The Figure 5 shows the original input sound with a mixed background noise, we can see in the figure of time domain, the noise is heavy and in a long term.

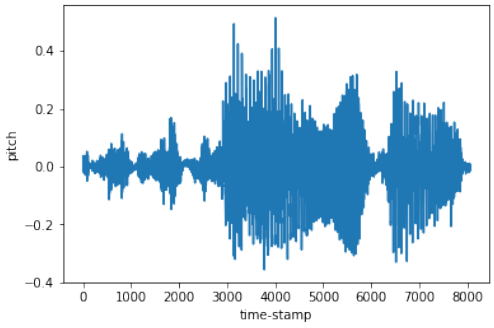

Figure 5. Original sound with noise (time domain).

Using our approach, first use STFT to get amplitude spectrum of original sound, next get the noise amplitude spectrum by using our pre-trained U-net, then subtract it with original amplitude spectrum, finally get the denoised speech voice by ISTFT from the denoised amplitude spectrum and the original phase. The results in the time domain are shown in Figure 6. We can see that, in the plots of denoised voice, there are many small fluctuations, which is known as the noise, are already moved from the noisy voice.

In summary, we can see from the results that our model and method can effectively denoise on speech.

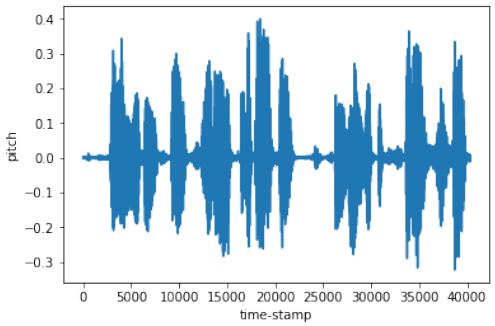

Figure 6. Denoise Results (time domain).

6. Conclusions

In this project, we have implemented a deep learning speech enhancement system for denoising environmental noise.The problem of audio noise reduction has been converted into an image processing problem by using the amplitude spectrum representation of the sound, which simplifies this problem. The noise to remove has been modelled by a U-Net, a Deep Convolutional Autoencoder with symmetric skip connections. After training from the dataset, the network is able to model 10 classes of environmental noises. This approach is imperfect since we only use the noisy amplitude spectrogram in U-net, this might decrease the total performance in particular cases. But still, this approach is robust enough and able to reuse for many voices and environment noise configurations. In furture work, we are going to try some new datasets which includes more kinds of noise to test the ability for the approach to reuse for other noise.

7. Contributions

Haoyang Ding: Finish the training part and write the Abstract, Project progress report

Shang Wang: Finish the data generation part

John Xie: Finish the prediction part

Yu Wang: Write the final report

References

[1] E. M. Grais and M. D. Plumbley. Single channel audio source separation using convolutional denoising autoencoders. pages 1265–1269, 2017.

[2] K. Han, Y. Wang, D. Wang, W. S. Woods, I. Merks, and T. Zhang. Learning spectral mapping for speech dereverberation and denoising. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(6):982–992, 2015.

[3] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde. Singing voice separation with deep u-net convolutional networks. 2017.

[4] K. J. Piczak. Esc: Dataset for environmental sound classification. pages 1015–1018, 2015.

[5] M. Portnoff. Time-frequency representation of digital signals and systems based on short-time fourier analysis. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):55–69, 1980.

[6] D. Rethage, J. Pons, and X. Serra. A wavenet for speech denoising. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5069–5073, 2018.

[7] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. pages 234–241, 2015.

[8] X. Zhang and J. Wu. Denoising deep neural networks based voice activity detection. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.