Selective Hearing

Samuel Schmidgall

December 6th, 2018

Abstract

Hearing loss, Schizophrenia and ADHD are all irreversible handicaps that make it difficult for a surprising large population of affected individuals to listen and focus in on a single voice in a noisy environment. What if it were possible to cut out the noise out of the environment so that a given individual could solely hear the speaker? The goal of this work was to use Data Mining techniques with the goal of removing extraneous noise from an environment where an individual is speaking. In this project, podcast data is taken from the MIT podcast dataset, and sinusoid waves of varying frequencies are combined with the original dataset to serve as noise files. The dataset for this project contained 51,033 wav files which totaled to 41.94 GBs. Using the noise files as the input data and the frequency of the sinusoid wave as the expected value, three statistical models were trained with the goal of predicting which frequency of noise was added to the data element in question; a Support Vector Machine, an Artificial Neural Network, and a Convolutional Neural Network. The Artificial Neural Network showed the best results with an average error of 21 hz on the training set and 28 hz on the test set.

Introduction

Audio Noise Processing was originally dominated by professionals in Electrical Engineering departments, using filtering techniques that required a lot of information about the noise before it was removed; this domination has been subsiding as Data Mining techniques show more and more promise. In this paper we explore using Data Mining techniques to remove high frequency noise from audio data in the form of human speech.

Problem Statement

3.5 million people are affected by Schizophrenia in the U.S. Likewise, 6.4 and 37.5 million people are affected by ADHD and Hearing Disabilities in the U.S. alone[8, 9, 10]. What do all of these disabilities have in common? Well, each one of these handicaps makes it difficult for an extremely large population of affected individuals to focus in on a single voice in an environment filled with noise. These aren't the only places which noise reduction are necessary, but also in applications such as live news broadcasting, where noise reduction techniques must be made real-time, music restoration, where high frequency noise is generated when converting from analog to digital, and finally in hearing aids where high frequency noise makes it challenging for the user to focus in on a single voice. In this paper we will look at independent research focused on removing high frequency noise from audio data.

Literature Review

A surprising number of people throughout the data mining literature have worked on audio noise removal with varying degrees of accuracy – mostly containing mediocre results. A wide variety of papers looked at different types of filtering, such as Butterworth, Chebyshev and Elliptical[1] which each rely on really sophisticated techniques that sadly don't utilize data mining. Another common approach to solving the problem is by introducing the notion of a Wavelet Filter, which essentially manipulates the amplitude of a wavelet according to their frequency[2], though this too doesn't utilize any data mining. Finally, in the more modern literature there have been approaches toward removing audio noise using machine learning, more commonly using artificial neural networks.

The first interesting paper that caught my eye due to its relevance to my project, and interests, was titled "Removing Noise from Speech Signals using Different Approaches of Artificial Neural Networks". In this paper, Dr. Al-Allaf from the University of Jordan constructs and tests four different types of artificial neural networks with the goal of observing and comparing the different results, which is what originally inspired me to set up and try multiple artificial neural network models. In her paper, Dr. Al-Allaf identifies three different broad methods for removing noise from audio data; namely: filtering, noise subtraction and space mapping. Filtering is a technique that is primarily used for speech enhancement. Though there are many different filtering algorithms, the primary theme is that they use a set of data collected over time to estimate a noise probability distribution. What makes noise subtraction techniques unique is that the noise and speech are assumed to be independent of each other. In this case, the noise power-spectrum is estimated and then subtracted from the audio. The final technique is called space-mapping. In space-mapping, the goal is to transform noisy speech into clean speech by mapping each time-step in the noisy data to a clean time-step. In this project I had decided to use noise subtracting since noise was being added to the clean dataset, therefore they were naturally independent.

The next paper that I read which inspired a good portion of my work, and was mentioned in my video presentation, was Se Rim Park's paper[3] titled 'A Fully Convolutional Neural Network for Speech Enhancement'. Park's paper was one of the first noise filtering papers that I came across where the goal of the project aligned near perfect with mine. In fact, the goals and applications for Park's paper matched mine quite well, which was really exciting. For Park's project, their team implemented both a fully convolutional neural network and a convolutional neural network that automatically removes redundant connections to minimize the amount of space the network takes up so that it can easily be embedded into devices such as hearing aids. In the end Park demonstrated that convolutional neural networks can reach similar or better performance than state of the art recurrent neural networks, while also taking up much less memory. Park's usage of convolutional neural networks is what inspired me to use them in my project.

Methods and Techniques

Audio Data is amazingly tricky to work with. The first step after getting all of the data is converting the MP3s into wav files because Python doesn't fancy MP3s quite as much as it does wav files. Once all of the MP3s are converted to wavs, to actually get decent information out of the file, you have to run a Fast Fourier Transform on the parsed wav file. A Fourier Transform decomposes a function of time, in this case the audio data, and converts it to a function in the frequency domain. The reason that this is important is because audio noise typically deviates from the normal frequency patterns seen in previous un-noised data; this would be undetectable if the audio data were still in the time domain because all the different frequencies are combined into one massive wavelet. More practically, the Fast Fourier Transform is an algorithm that computes the Discrete Fourier Transform, in our case because audio data is inherently stored as discrete data. The Discrete Fourier Transform is generated by decomposing a sequence of values into components of different frequencies[7]. This is the format of the data that is input into the machine learning models.

Using the Fast Fourier Transform to transform all of the audio data to a function of the frequency domain, I generate a training matrix. Alongside the training matrix I generate an expected value vector, where the expected value for the element in question is the frequency of the sine wave that was added to it, which we know.

As far as the statistical models go, as mentioned before, there are three different types that are being used in this project; the Support Vector Machine, Artificial Neural Network, and Convolutional Neural Network. The Support Vector Machine is a Support Vector Regressor using an RBF filter from the Scikit Learn library. Likewise, the Artificial Neural Network was generated using the Scikit learn library, which is likely the reason that it outperformed the CNN. The Convolutional Neural Network was actually a 13-layer custom built network using the Python library titled Keras. The Convolutional Network starts out with a linear convolution layer with a size 3 kernel and using 'same' padding to retain the input size. The next few layers are, a Leaky ReLU with an alpha of 0.1, a Max Pooling with a size 2 kernel and same padding, then a Convolution layer with a size 3 kernel and all of the same parameters as the previous. This same pattern is repeated once more. Afterward, a flatten function layer is called followed by a Dense layer, a Leaky ReLU and then finally another Dense layer.

The development of this Convolutional Neural Network was mostly on intuition and trial and error, which I would change if I had more time to work on this project. I would assume that given more time to try different layers in different combinations, the Convolutional Neural Network would outperform the Artificial Neural Network.

Discussion and Results

Dataset

Considering how bafflingly high dimensional audio data is, it quickly becomes clear that a large dataset must be utilized in any given project. The main data set that I used was the MIT Podcast Dataset[4]. The original usage of this dataset was for generating audio summaries of the podcasts. The reason that I chose this data set was actually for quite a few reasons. The main reason that I chose this data set was because it was a large data set. Before tampering with the data, there were around 5,000 podcasts which took up about 4 GBs of space. Now, while this isn't the largest dataset ever, it becomes quite large a bit later down the line. The next reason that I chose this dataset was because podcast data is very clean; there is next to no noise in a given podcast episode. This makes podcast audio the ideal candidate for a project where you are adding noise and then trying to identify the frequency of the added noise; if there were noise in the original data then that would make the test set validation much less accurate.

After I had all of the podcast data the next step was to figure out how I was going to get noise data and how I was going to format my input data and expected value utilizing the podcast dataset. The type of noise wavelets that I decided to add into my podcast dataset were sine waves of varying frequencies between 440 and 1440; all frequencies which are audible to humans. Each podcast had 100 different sine wavelets added to them to form a new podcast, which in turn multiplied the size of my dataset by a large factor. The new dataset contained 50,000 audio files which totaled to around 40 GBs of space. Now the podcasts that contained the noise were able to serve as the model input, and the sine frequency that was added to the podcast could serve as the expected value. Then, once the frequency was retrieved it could be subtracted from the audio file.

Evaluation Metrics

To evaluate the performance of a network after it had been trained, a metric must be specified. In the case of my project, the metric I had defined was a simple L1 distance from the output frequency to the expected frequency. For example, if the model output a frequency of 17 and the expected frequency was 23 the distance would be |17-23| equals 6.

Experimental Results

After training each of the three models, the Support Vector Machine, Artificial Neural Network, and Convolutional Neural Network, to evaluate their performance I computed each models' average L1 distance away from the training set, which contained 1000 unseen test samples. The performance varied for each model with the Artificial Neural Network mildly outperforming the Convolutional Neural Network, and the Convolutional Neural Network outperforming the Support Vector Machine by a wide margin. The highest performing network, the Artificial Neural Network, had an average L1 distance of 21 hertz on the training data set and only 28 hertz on the test data set. The second best performance was the Convolutional Neural Network, which has an average distance of 33 hertz on the training data set and 38 hertz on the test data set. The Support Vector Machine did not perform well likely due to the high dimensionality with an average distance of 40 hertz on the training data set, and 49 hertz on the test data set.

Conclusion

Audio data is without a doubt the most difficult data that I have ever had to deal with. Even to get it into a semi-manageable state takes an enormous amount of work and research. On top of all of that, once it is in a decent state, the dimensionality is beyond comprehendible. I was surprised to see how well the models ended up performing after I had added the convolution step to the preprocessing compared to how they were performing before. Overall, I do think that this project was a bit too ambitious to do alone in less than 6 weeks, but I am more than pleased with the results that I got in such little time. Otherwise, I am excited about how well my results turned out and am looking forward to continuing this work into the Spring semester.

Directions for Future Work

There are definitely many different directions that this research could head in; audio processing in machine learning is still in its infancy. I suppose the trivial direction would be attempting the same goal that I had for this project, just with different models. I would definitely like to recreate Se Rim Park's machine learning model, which utilizes space mapping rather than noise subtraction. Another addition that I would like to make in the future would be to try removing real noise[6] from the audio data rather than just generic sine waves, which would be a nice challenge to try alongside Park's model. Another interesting direction that I could head would be attempting to work on a model that attempts to solve the cocktail party effect[5], which is relevant with the idea in mind that you could train a network to treat voices other than the target voice as noise.

As far as applications go, one direction that I could head in would be trying to imbed my model into a hearing-aid and seeing how well it performs. There would have to be real-time modifications made to the algorithm, and if it doesn't perform well enough, then I could work on reducing the redundant connections in the algorithm similar to [3].

[1] http://www.iraj.in/journal/journal_file/journal_pdf/6-162-1440572779132-135.pdf

[2] http://www.ijcstjournal.org/volume-2/issue-5/IJCST-V2I5P28.pdf

[3] Se Rim Park: "A Fully Convolutional Neural Network for Speech Enhancement", 2016; arXiv:1609.07132.

[4] Damiano Spina, Johanne R. Trippas, Lawrence Cavedon, Mark Sanderson Extracting Audio Summaries to Support Effective Spoken Document Search Journal of the Association for Information Science and Technology. 2017.

[5] https://www.audiology.org/news/cocktail-party-effect

[6] Jort F. Gemmeke and Daniel P. W. Ellis and Dylan Freedman and Aren Jansen and Wade Lawrence and R. Channing Moore and Manoj Plakal and Marvin Ritter: "Audio Set: An ontology and human-labeled dataset for audio events". 2017; Proc. IEEE ICASSP.

[7] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1447887

[8] https://www.cdc.gov/ncbddd/adhd/data.html

[9] https://www.nidcd.nih.gov/health/statistics/quick-statistics-hearing

[10] https://www.nimh.nih.gov/health/statistics/schizophrenia.shtml

https://www.researchgate.net/publication/282446599_Removing_Noise_from_Speech_Signals_Using_Different_Approaches_of_Artificial_Neural_Networks

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html