Sound Classification: A Self-Learning ML Model for the Deaf

Sumer Shinde and Sathvik Yadanaparthi, Shrewsbury High School

Abstract

On a daily basis, humans heavily rely on their ears to alert them of acoustic cues in the environment, which prompts awareness of changes in their surroundings. These cues help people to react to those changes, potentially helping them to avoid dangerous situations. But for the 450 million people suffering from disabling hearing loss, this basic ability is impossible (*Deafness*, 2018). Consequently, they are constantly placed in a position of uncertainty, which possibly puts them in harm's way. This could be resolved if they were informed of the dangers that surround them and were able to react and protect themselves. In order to help, our goal is to create a device that can accurately detect sounds, even in busy urban areas, and differentiate between dangerous and everyday loud sounds, while adapting and self-learning its classification. First, with data from an UrbanSounds dataset, background noises will be filtered out of each recording with a software called Audacity, making the sound more apparent. Then we will use fast fourier transform (FFT) to create a periodogram, which then can be used to calculate the mel-frequency cepstral coefficients (MFCC). We use these values to train CNN, while implementing Hummingbird to improve performance. We will deploy our ML model onto the Raspberry Pi and create a program which loops the ML to make it continuously sample. Finally, if there is a situation where two sounds overlap, we will create a program that will use a confusion matrix to choose which sound to notify the user.

Problem

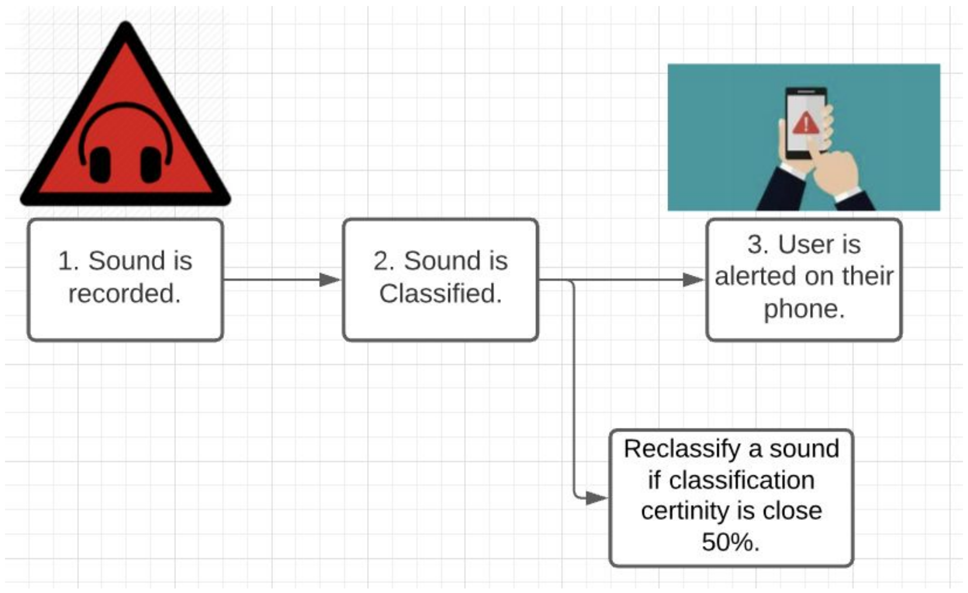
The average person easily overlooks the ability to hear, but for approximately 5% of the world, hearing is considered a privilege. People who are hearing impaired are constantly placed in danger due to their inability to be aware of their environment. Those who are deaf or hard-of-hearing have been found to have a higher chance of getting injured, simply because they were not aware of the danger such as a gunshot (Correll, 2020).

Current Work

There have been a number of studies conducted on sound classification for the deaf from many different approaches, but every study has had significant flaws. Ma et al. developed an adaptable system for sound classification by calculating the MFCC of feature extraction of audio files and used the Hidden Markov Model (HMM) algorithm as a classifier. Their testing accuracy was reported at 92%, but the accuracy drastically dropped to 35% when tested in the real world. Suh et al., also showed the same trend in accuracy, but used an ensemble model and Long Short-Term Model (LSTM) for classification and used a vibrator connected to a Raspberry Pi to alert the user. Testing their models showed a 98% accuracy rate in both, but when presented in a real world situation the accuracy dropped to 72%. Though Su et al. took a different approach using Local Discriminant Bases technique where the distinction of time-frequency subspaces for ambient sounds is defined, its accuracy while testing showed 81% but dropped to 28.6% when applied to the real world. In all of these studies, accuracy was relatively high during testing in a controlled environment where classification did not have to account for the background noises and other external factors. But when applied to the real world, they produced unreliable classifications which ultimately failed to make the deaf user aware of his or her surroundings. Park & Lee and Fanzeres et al. used similar approaches to conduct studies on environmental sound recognition, but their classification time was 5.2 seconds and 3.6 seconds respectively. Though their models showed an acceptable accuracy of over 80%, the classification time is far too long. This gap is significant, as even a second wasted can put the deaf user in danger.

Solution

We are proposing to create a device that can alert the deaf to the specific sounds in their environment regardless of background noises in order to make their lives easier and safer. This device will be able to accurately detect a sound and classify the sound in less than one second. This would allow for the user to react to a certain sound in time and potentially avoid being in a harmful situation.

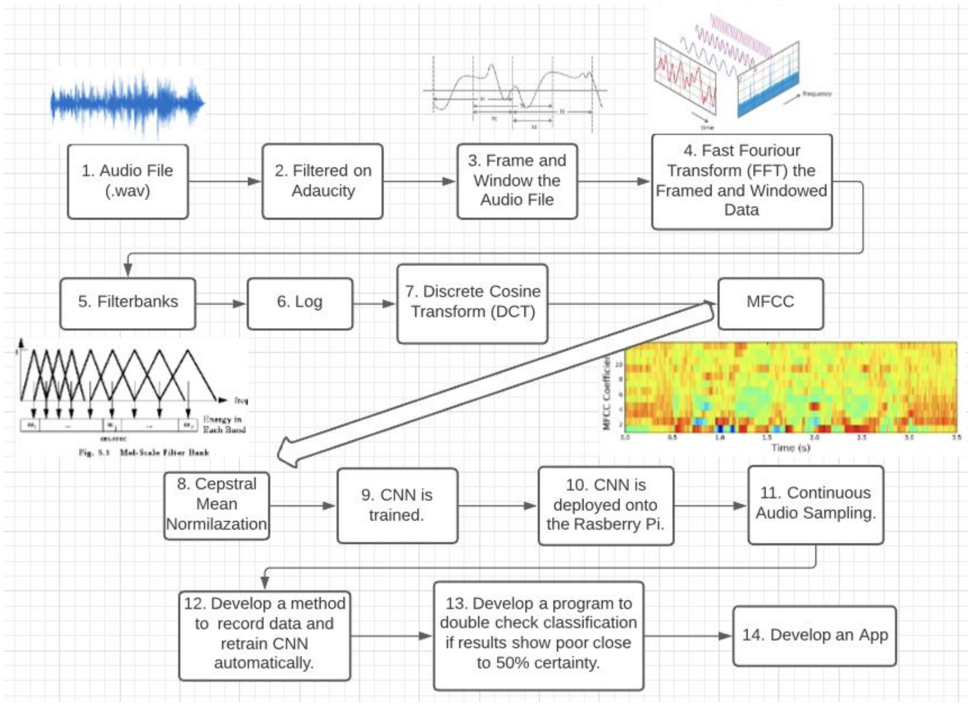

1. Sound is recorded.
2. Sound is Classified.
3. User is alerted on their phone.
Reclassify a sound if classification certinity is close 50%.

There have been studies aimed to solve this problem by also developing an audio classification system, as mentioned in the current work section. Our project utilizes a number of methods to improve the accuracy and classification time, both of which aspects were the defects of the existing studies. To improve the accuracy of the model, we will filter all the background noise in the initial data before developing our classification system. For example, if the sound of a dog barking from the dataset is recorded in a city, but in the real situation the dog is barking in a quieter area, this would decrease the accuracy of the model because it is not only looking for the dog barking, but also the sounds of bustling streets and roaring cars which do not exist in the countryside. But by only capturing the dog barking when training the neural network, the ML model will always classify the dog regardless of the situation.

We will create an adaptive system by saving audio samples of when a sound is classified in the real world and retrain our classification system based on this new data. This would allow for the model to constantly adapt and improve itself based on real world data. For example, if a dog is barking and it is classified, then a 20 second sample will be recorded and saved onto the Raspberry Pi. Therefore, the ML model can constantly improve its accuracy by adding more real world data. This will allow for the CNN to learn and provide a more accurate classification next time when a similar situation is presented. We will also write a program to reclassify a certain audio sample if certainty rates between two or more sounds are very close to each other. This method accounts for a situation where more than one sound is heard so that both sounds can be brought to the awareness of the user, which further increases the accuracy of the device. Lastly, to decrease the classification time, we will be implementing Hummingbird, a library for compiling trained traditional ML models into tensor computations, which will allow us to limit the classification time to less than one second.

Approach

In order to create the classification system, we will develop a convolutional neural network (CNN), a deep learning neural network designed for processing structured arrays, using the Urban Sounds Database as our dataset (Wood, n.d.). Since the raw data contains a large amount of redundant information, it would be highly inefficient to use the data in its original format; therefore, we will calculate the Mel Frequency Spectrum Coefficients (MFCC), a short-term power spectrum of a sound, of the data in order to convert the raw data into a simplified form. Using the spectrum coefficients (MFCC), we will train the CNN model. After training a neural network, we will deploy the model onto the Raspberry Pi to classify sounds live by using an omnidirectional microphone. Lastly, we will develop a method which will save an audio sample of a sound in the real world when recognized, and the neural network will be


1. Audio File (.wav)
2. Filtered on Adacuity
3. Frame and Window the Audio File
4. Fast Fouriour Transform (FFT) the Framed and Windowed Data
5. Filterbanks
6. Log
7. Discrete Cosine Transform (DCT)
MFCC
8. Cepstral Mean Normilazation
9. CNN is trained.
10. CNN is deployed onto the Rasberry Pi.
11. Continuous Audio Sampling.
12. Develop a method to record data and retrain CNN automatically.
13. Develop a program to double check classification if results show poor close to 50% certainty.
14. Develop an App
Fig. 5.3 Mel-Scale Filter Bank
MFCC Coefficients
Time (s)

retrained, based on the new data collected, to increase the accuracy of the classification.

#. - Correlates to the steps above

1. We will obtain the data as .wav files from Urban Sound Database (*Urbansound8K Dataset*, n.d.). **2.** In order to assure the accuracy of the ML model regardless of which external factors are presented, we will filter the audio using the Noise Reduction feature on Audacity (*Noise Reduction*, n.d.). This would remove most of the background noises in the data so that the CNN will be able to more accurately detect a certain sound in any given situation. After filtering out the background sounds from the data, we will compute the MFFC in order to simplify the raw data. **3.** First, we will split the audio signal into short time frames because audio signals constantly change. With this constant change, when we perform the Fast Fourier Transform

(FFT) on the entire signal, we will lose the variations in frequencies, which are the determining factors (*Fast Fourier Transformation FFT - Basics*, n.d.). So by framing we are able to assume that frequencies will not change over short periods of time, and we can just run FFT without losing the frequency contours. After splitting the signal into frames, we will apply the window function. **4.** Next we will use the Fast Fourier Transform (FFT) on each frame to convert the data from a time domain to the frequency domain which allows us to determine which frequencies exist in each frame and calculate the power spectrum (periodogram) (*Fast Fourier Transformation FFT - Basics*, n.d.). **5.** We will use the Mel filterbank to split apart the periodogram into separate clumps of periodogram bins and combine them to estimate the amount of energy in each of the frequency regions (*Filterbank Analysis*, n.d.). **6.** Next we will log the filterbank energies in order to scale down the energies so we will be able to determine any variations in them. **7.** Lastly, we will calculate the Discrete Cosine Function (DCT) of the log of filterbank energies to decorellate the energies because the filter banks overlap each other which would make them correlated (*Discrete Cosine Transform and Inverse DCT*, n.d.). This would give us the cepstral coefficients (MFCC) of the data which we can use to train our CNN. **8.** To balance the spectrum and improve the Signal-to-Noise (SNR) we will compute mean normalization on all the frames. **9.** We will train the convolutional neural network based on the MFCC (Karkare, 2019). We will also implement the Hummingbird program into our model so that we can improve its performance.

10. We will deploy our ML model onto the Raspberry Pi, and using Tensorflow Lite we will be able to run the model efficiently (inferencing is performed in less than a second) on a device with limited capabilities such as the Raspberry Pi. **11.** We will create a program to loop the ML model so that it will continuously sample without missing any sounds. **12.** We will develop a method for an audio sample of 20 seconds to be saved and stored on the Raspberry Pi when a sound is classified. Using this data the neural network will be retrained automatically with the addition of the new data when the device is not in use. To accomplish this goal, we will use the Google AI Platform to train our model because the Raspberry Pi does not have the GPU or CPU power to handle this function. We develop a method to upload the sample audio files on the Raspberry Pi into the AI Platform when the device is connected to WiFi then will retain the CNN and compile it and send it back to the Raspberry Pi. **13.** We will write a program on the Raspberry Pi to reclassify a certain sound if results show 50% certainty for a certain sound. This will slightly increase the alert time but since classification takes less than 1 second, it is unnoticeable. If the certainty rates between two sounds remain close to 50% over the course of two classifications, then the user will be alerted of both sounds. Below is an example of a situation when this would occur.

	Certainty Rate
Dog Barking	55%
Gunshot	45%

14. Lastly, we will develop an app which connects the Raspberry Pi to the user's phone. This will allow for notifications/alerts to be sent to the user if a sound is recognized, and it will create an interface for the user to interact with in order to train the ML model to recognize the user's name.

Resources

The materials we will need are included in the table below in the Project Budget Section. We would specifically like mentorship from the THINK team on developing a program to automatically save 20 second data samples and uploading it into the Google AI Platform. We also would appreciate any other general advice and feedback on improvements we can make to the project by the THINK Team or MIT professors. Additionally, we hope to have access to a larger and more defined audio dataset from MIT.

Goals

For our project, we would like set a couple of milestones:

1. **Develop a method which can filter out background noises from the prerecorded dataset.**
2. **Calculate the cepstral coefficients (MFCC).**
3. **Train the neural network and deploy onto Raspberry Pi.**
4. **Implement continuous audio sampling.**
5. **Develop a method to automatically save sample recordings when a sound is detected and retrain the model including the newly acquired data.**
6. **Develop a program to reclassify a sound if results show great uncertainty.**
7. **Develop an app for device-to-user interactions.**

After achieving these goals, in order to gauge accuracy, we will test the CNN both in controlled and uncontrolled environments. An example of a controlled environment would be a sealed room where no sound enters and all sounds are controlled by us for testing purposes. An uncontrolled environment would be in a real world situation, such as cities, schools, or houses so there is background noise, for which the device must account. Our project specifications include having an accuracy of above 90% to ensure that the program works as expected. Similarly, we would like to also have a F1 score of above the threshold of 0.9. A F1 score is a function of Precision and Recall, which are also other forms of calculating accuracy (Shung, 2018). The function is calculated as below:

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

(Shung, 2018)

As partners, we have designed a plan to work together safely during the pandemic; we plan to work in tandem over Zoom instead of splitting the work into parts, as each part heavily relies on the previous one. We will only physically meet when testing our CNN, but we plan to abide by the social distancing rules.

Risks

Some issues may arise when it comes to implementation:

1. **The accuracy of the model is under 90%.**

Unfortunately, there is a possibility that the accuracy of the model may be lower than we expected due to any number of reasons. To fix this issue, we will train our CNN with additional datasets provided by Audioset or we could increase the number of epochs during the initial training of the model.

2. **The neural network is not automatically retraining itself with the new data found.**

If the neural network is not retraining itself with the new data obtained, we will allow for the user to manually retrain the CNN on the app.

3. **The approach to determining the MFCC is unsuccessful/inaccurate.**

If the method to calculate the MFCC is unsuccessful/inaccurate, we will instead use a Linear Prediction Coefficients (LPC) as a means to extract the audio features.

Timeline

1. 3/1/2021 - Filter out sounds from dataset
2. 3/14/2021 - Calculate the spectrum coefficients (MFCC)

3. 3/21/2021 - Train the neural network and deploy onto Raspberry Pi.
4. 4/4/2021 - Develop a method for continuous audio sampling.
5. 4/18/2021 - Develop a program to record 20 seconds of a sample when one is heard
6. 5/1/2021 - Develop program to double check classification results
7. 6/1/2021 - Testing Complete and model will be fully trained and be as accurate as possible

In order to document the implementation process between these deadlines, we will log our progress in a scientific notebook, making notes about our developments and any problems we encounter along the way.

Current Progress / Funding

We have already developed a working sound classification ML model, but it was created with our own dataset. This could only classify sounds when prompted to and could not automatically retrain itself. What needs to be done is developing a method for filtering out unwanted background noises for the primary dataset, implementing continuous audio sampling, and automatically saving audio samples from the real world and retraining our model from it. With funding from the MIT THINK Scholars Program, we will be able to get all of the hardware we need in order to apply this device to the real world and make it user ready instead of only creating an adaptable sound recognition software.

Project Budget

We chose the 8GB RAM version of the Raspberry Pi to process the vast amount of data and run our ML model faster. We also specifically chose a high quality, medium range, omnidirectional microphone to pick up sounds from all directions. The Raspberry Pi 4 Case contains a rechargeable lithium ion battery which allows for the device to be portable and provides cooling fans for the Raspberry Pi to protect it from overheating. Additionally, we will need funding for using the Nvidia Tesla K80 GPU on the Google AI Platform.

Item	**Cost**	**Link**
Rode VideoMicro Compact On-Camera Microphone with Rycote Lyre Shock Mount	$59.00	Link
Raspberry Pi 4 Model B/8GB	$75	Link
Raspberry Pi 4 Case with Fan, 5 Hour Internal Battery, Mini OLED Screen and Integrated Speaker	$99.95	Link
PNY 128GB Elite-X Class 10 U3 V30 microSDXC Flash Memory Card	$24.99	Link
Nvidia Tesla K80 GPU	$0.45 per hour	Link

Interest

As students interested in all areas of computer science, we ultimately wanted to use our computer science skills to develop a device for the betterment of humanity. We are specifically interested in helping the hard of hearing because we both have family members who suffer from disabling hearing loss. In the past, we have created a device which detects dangers in an environment based on the decibel level. This device would alert the user to any loud sounds that may be caused by something dangerous. For example, if a gunshot were to go off near the deaf user, the device would alert the user by vibrating and prompting the user to call the police or a loved one. But we soon realized that depending on the decibel level was inaccurate because dropping the device onto the ground would create the same level of sound as a gunshot. After rethinking our project, we wondered: why not just detect the specific sound itself? After some research, we discovered that there were already some advancements in sound classification, but these projects were proven to be ineffective in real world scenarios. Therefore, we decided to improve upon this idea by creating an accurate version.

Qualifications

We both are very interested in computer science and have been pursuing our passion through coursework, jobs, and extracurricular activities since our freshman years. We both have experience in Python and Java as well as basic proficiency with Tensorflow. Some skills we may need in the future are app development and proficiency in audio filtering.

References

Abesser, J. (2020, March 16). A Review of Deep Learning Based Methods for Acoustic Scene Classification. *applied sciences*, 16. https://www.mdpi.com/2076-3417/10/6/2020/pdf

Converting a Neural Network for Arm Cortex-M with CMSIS-NN. (n.d.). Arm Developer. https://developer.arm.com/solutions/machine-learning-on-arm/developer-material/how-to-guides/converting-a-neural-network-for-arm-cortex-m-with-cmsis-nn/next-steps

Correll, R. (2020, July 7). *Challenges That Still Exist for the Deaf Community*. verywell health. https://www.verywellhealth.com/what-challenges-still-exist-for-the-deaf-community-4153447

Davis, S. B., & Mermelstein, P. (n.d.). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. 24. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.462.5073&rep=rep1&type=pdf

Deafness. (2018, March 15). World Health Organization. https://www.who.int/news-room/facts-in-pictures/detail/deafness#:~:text=There%20are%20466%20million%20people,rise%20to%20over%20900%20million.

Difficulties the Hearing Impaired Face Every Day. (2018, September 14). Disability Experts of Florida. https://www.disabilityexpertsfl.com/blog/difficulties-the-deaf-face-every-day#:~:text=Studies%20reveal%20that%20deaf%20people,usually%20talking%20with%20a%20therapist

Discrete Cosine Transform and Inverse DCT. (n.d.). O'Reilly. https://www.oreilly.com/library/view/vlsi-digital-signal/9780471241867/sec-9.3.html

Fast Fourier Transformation FFT - Basics. (n.d.). NTi Audio. https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft

Filterbank Analysis. (n.d.). https://labrosa.ee.columbia.edu/doc/HTKBook21/node54.html

Karkare, P. (2019, February 15). *Convolutional Neural Networks — Simplified*. Medium. https://medium.com/x8-the-ai-community/cnn-9c5e63703c3f

Ma, L., Milner, B., & Smith, D. (2016, July). Acoustic environment classification. *ACM Transactions on Speech and Language Processing*, *3*(2). ACM Digital Library.

Malek, A. (2020, January 25). *Signal framing*. Ayoub Malek's Blog. https://superkogito.github.io/blog/SignalFraming.html

Mesaros, A., Hettola, T., & Virtanen, T. (2020). *Acoustic scene classification*. Detection and Classification of Acoustic Scenes and Events. http://dcase.community/challenge2020/task-acoustic-scene-classification#:~:text=The%20goal%20of%20acoustic%20scene,in%20which%20it%20was%20recorded.

Noise Reduction. (n.d.). Audacity. https://manual.audacityteam.org/man/noise_reduction.html

Park, G., & Lee, S. (2020, March 7). Environmental Noise Classification Using Convolutional Neural Networks with Input Transform for Hearing Aids. *Int J Environ Res Public Health*, *17*(7). NCBI. 10.3390/ijerph17072270

Shung, K. P. (2018, March 15). *Accuracy, Precision, Recall or F1?* Medium. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Snapshot of Deaf and Hard of Hearing People, Postsecondary Attendance and Unemployment. (n.d.). Gallaudet University. https://www.gallaudet.edu/office-of-international-affairs/demographics/deaf-employment

-reports#:~:text=Across%20all%20age%20groups%2C%20approximately,over%2065%20years%20of%20age

Su, F., Yang, L., Lu, T., & Wang, G. (2011, November). Environmental sound classification for scene recognition using local discriminant bases and HMM. *MM '11: Proceedings of the 19th ACM international conference on Multimedia*, 3.

Suh, H., Seo, S., & Kim, Y. H. (2018, December 8). *Deep Learning-Based Hazardous Sound Classification for the Hard of Hearing and Deaf*. IEEE. https://ieeexplore.ieee.org/document/8642632

Urbansound8K Dataset. (n.d.). Urban Sound Dataset. https://urbansounddataset.weebly.com/urbansound8k.html

Wood, T. (n.d.). *Convolutional Neural Network*. DeepAI. https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network