Kalin Kochnev, Rohan Menon, Jacob Yanoff

3D Audio Localization - a Hybrid Approach to Sound Positioning, Recognition, and Wearable Presentation

Abstract

Recent advancements in machine learning and image processing mean that the ability of computers to see has improved drastically in the last 10 years. While sound is a crucial part of how most people experience their environments, computer hearing has not seen the same advancements. We aimed to develop algorithms to locate audio signals within 3D space as well as classify them into several relevant categories. Additionally, we wanted to convey this information to a user via a wearable device. The final device uses cross-power spectrum phase analysis to determine the arrival angle of arrival based on two pairs of microphones and displays this information via a heavily modified baseball cap. It uses a small, brim-mounted OLED display to convey positional information to the user. We imagine that potentially, it could be used by a person who is deaf or hard of hearing to better understand their soundscape. The classification algorithm relies on an artificial neural network generated through supervised deep learning. The localization algorithm proved to be highly accurate, with an average error of 2.53% when determining the relative angle of a sound source. The machine learning algorithm is quite successful at identifying test data, exhibiting 84.6% accuracy, however, overfitting is still present and further optimization is required to make the algorithm applicable to less contrived data. While these algorithms perform well independently, combining their functionality poses a new set of challenges that we hope to address in future research.

Introduction

In the realm of synthetic sensing, computer vision has been the predominant focus of research over the last several decades. Humans, being a visually dependent species, have been naturally drawn to artificial sight, however, artificial hearing can be an equally important field in terms of the breadth and impact of its applications. For example, human-interface devices, spatially aware robots, and aides for people with disabilities can all see massive improvements through the development of more advanced sound sensing technology.

Three-dimensional sound localization, or the ability to identify the position of an audio source, is among the most important skills for an artificial hearing system. Combining localization with classification would allow such a device to gain complete understanding of the soundscape and navigate the world with new awareness.

Solution

Goals

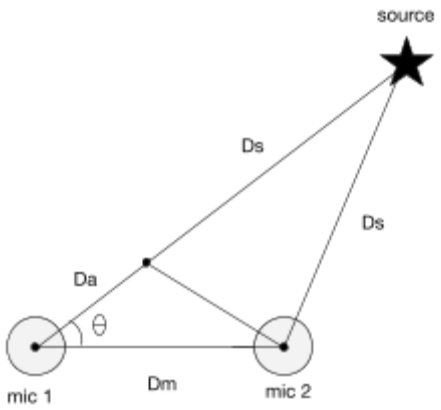
Through our research, we aimed to develop algorithms to both localize and classify sounds. They needed to be reasonably accurate while still being fast to compute, to achieve near real-time speeds.

These algorithms would eventually be moved to an embedded platform within a wearable device. This wearable needed to be light-weight, unobtrusive and have a visual interface that is easy to understand.

1. Sound Localization

We researched several different methods of localization, each of varying difficulties, but it became clear that one of the simplest method also produced the best results. We settled on using Cross-Power Spectrum Phase Analysis (CSP) to estimate the angle of arrival using pairs of microphones. Before finding out about this method, we attempted to come up with our own solution. Coincidentally, we derived the same equation that CSP uses.

Assume that an audio source is an arbitrary distance and angle away from a pair of microphones.



Given the difference in arrival time of the sound signal to each microphone, M_1 and M_2, we must calculate the source angle relative to the microphone array. In order to solve for θ, one must make the assumption that D_m is significantly less than D_s, and take the limit as D_s approaches infinity.

$$\lim_{D_s \to \infty} D_s - \sqrt{{D_s}^2 + {D_m}^2 - 2D_s D_m \cos(\theta)} = D_m \cos(\theta)$$

Because D_m is the hypotenuse of the right triangle in the figure, we can solve for θ:

$$\cos(\theta) = \frac{D_A}{D_m}$$
$$\arccos(\frac{D_A}{D_m})$$

We can then re-express D_A in terms of the time offset between two audio signals and the speed of sound

$$\arccos(\frac{V_{sound}\tau_{delay}}{D_m})$$

This simple equation is fundamental to sound localization not only in two dimensions, but can be extended to three dimensions.

1.1 Determining Signal Shift

Now we must calculate the time between two audio signals. The simplest way is determining a common artifact in two audio signals and measuring the delay, such as relative maximums. However, this method is prone to external interference from extraneous audio sources.

We determined that the cross-correlation of two signals was a good candidate for determining shift.

A cross correlation function, $f\{s_1(t),\ s_2(t)\}$, takes two signals and quantifies how similar they are to one another. Alternatively stated, this function is maximized when the signals are most similar. Shifting one of the signals across various times, t_{delay}, we can determine the point where the signals are most similar.

Here is an example of a simple equation that we could have implemented, albeit once converted to a discrete form. For simplicity, we consider any noise signal to be included as a part of the microphone signal.

$$C(\tau_{delay}) = \int_{-\infty}^{\infty} \overline{s(t)}\, s(t + \tau_{delay})\, dt$$

After running some simple tests, the cross-correlation algorithm was effective at determining the delay between two signals, but was well outside of our computation time constraints. We used Python to implement the algorithms, and while it is not as fast as C, we were confident that was due to the implementation of cross-correlation.

1.2 Speeding up calculations

The similarity between cross-correlation and convolution can be exploited for efficient computation. Convolution describes how the graph of one function is modified by the other, while cross-correlation shows how similar signals are. The main difference between the two is that cross-correlation has one of the signals reversed in the time domain. By manipulating the convolution theorem:

$$\{s_1 \star s_2\}(x) = \int_{-\infty}^{\infty} s_1(t) s_2(t) dt = F^{-1}\{S_1 \cdot S_2\}$$

We can reverse one of the signals, or take the complex conjugate of one of the Fourier transforms to get the same effect.

$$s(-t) = \overline{S(w)}$$
$$Cross\ Correlation = F^{-1}\{S_1 \cdot \overline{S_2}\}$$

1.3 Computational Complexity

By default, the computational complexity of brute force autocorrelation is O(n^2), but the complexity of the Fourier Transform definition is O(nlog(n)), which is almost an order of magnitude faster.

1.4 Interpreting the Cross-Correlation

The equations found earlier are used for a single pair of microphones, but can be expanded to any number if one can consolidate the differences between their readings. One pair's readings allow for localization from 0 to 180 degrees. The wearable uses two pairs, on opposite corners, to localize a full 360 degrees.

Issues arise when the source approaches, or becomes collinear with a pair of microphones. The time delay between incoming signals approaches the maximum possible delay of Dm/Vs, which results in volatile cross-correlations since the adjacent cannot exceed the hypotenuse.

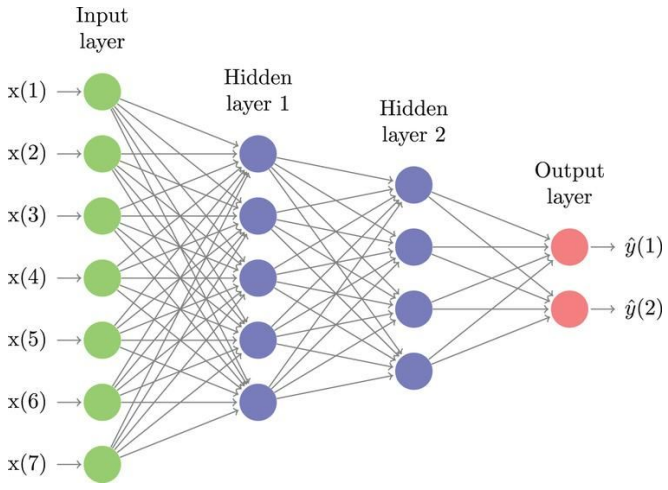
2. Audio Classification

Along with localizing sound inputs using CSP we aimed to recognize and classify a range of common urban sounds to display to a user useful information about both the location and identity of sounds occurring around them. Audio signal classification (ASC) is the act of extracting features from audio signals and using those features to determine to which of a number of classes that audio belongs. The features and sorting method used can vary greatly depending on the signals and mission of the program. The goal of ASC research in many ways is to create systems that can emulate humans ability to identify sounds and potentially improve upon it.

Our goal was to continuously classify audio clips into one of 10 categories: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun hot, jackhammer, siren, and street music. The choice of these specific categories was the accessibility of 8,732 labeled clips of less than or equal to 4 seconds from the audio dataset UrbanSound8k. This dataset is composed of crowd sourced audio clips that have been labeled with their respective category.

2.1 Machine Learning and Neural Networks

Machine Learning (ML) is a subset of Artificial Intelligence (AI) in which a computer learns in a very similar way to a human being. By analyzing its past experiences, machine learning is able to improve upon its understanding of a given task. Supervised learning is a form of machine learning in which input data, also known as features, and output data, also known as labels,are provided to the computer and the system attempts to find rules to solve any piece of input data. A neural network is a popular type of supervised learning in which a network of layers is constructed consisting of a number neurons. Similar to the human brain, these neurons accept input signals and modify those signals before passing it on to the next layer of neurons. To train a neural network, an optimizer changes the weights of each neuron and evaluates how well those changes worked. These changes happen in steps called epochs. This is done many times, typically hundreds or thousands, until the weights are able to produce the desired output given the input data.



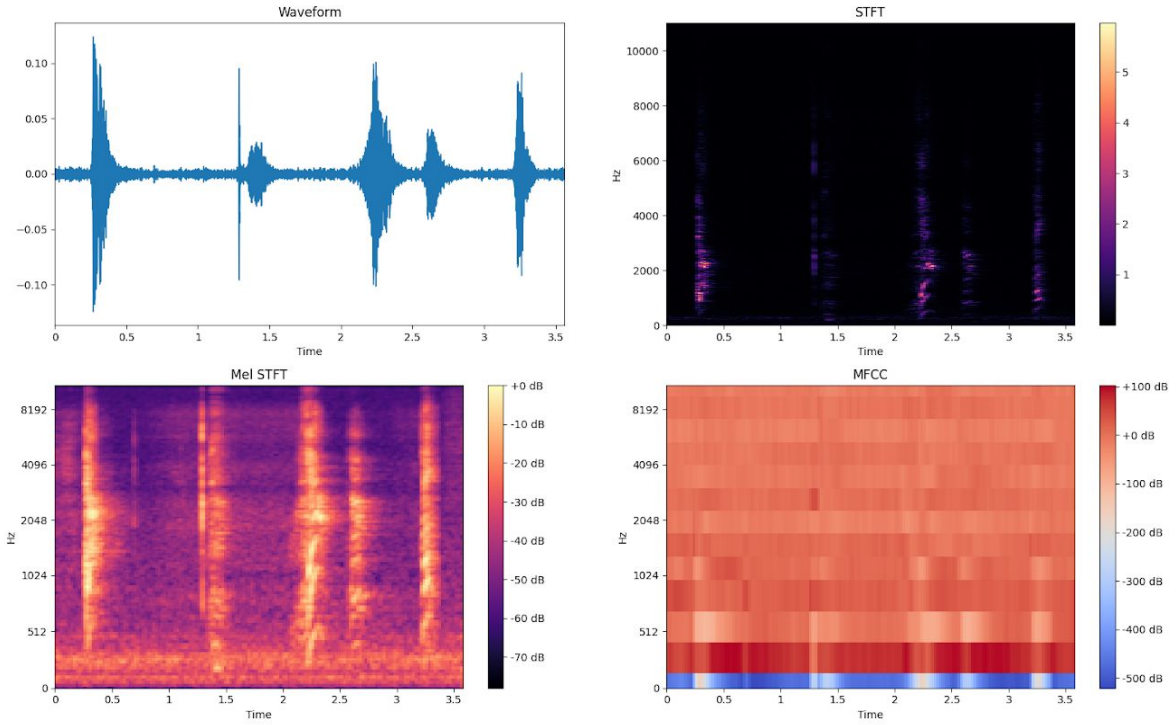
2.2 Feature Extraction

There are many popular features used for ASC. The best features vary widely depending on the use case. In our instance, we needed to classify short clips of audio with varying salience. Therefore, the features must represent significant information about the identity of the primary sound while excluding background noise. Furthermore, the feature must be a compressed form that is easy for the embedded processor on the wearable to compute in real time. Finally, the feature should be perceptually relevant, emulating the way humans perceive and understand the soundscape as the device will be used by a person and should be useful to their daily life. After significant research, it was determined that Mel Frequency Cepstral Coefficients (MFCCs) would be the most applicable feature for our deep learning model. MFCCs are calculated based on the Short Time Fourier Transform (STFT), wherein a Fast Fourier Transform (FFT) is computed over a range of intervals to demonstrate how the phase content of a signal changes over time. After the STFT is computed, the log spectral coefficients are mel-scaled. The mel-scale is a perceptually relevant frequency representation invented in the mid 20th century to analyze music. In other words, a one octave increase in the mel scale is an equal shift throughout the frequency range, this is obviously not the case for typical Hz representations.

These mel-scaled coefficients are then converted into a cepstrum using a Discrete Cosine Transform (DCT). The MFCCs can be calculated using the following equation where K is the number of filter banks, S_k is the mel scaled spectrum after passing through the kth mel filter bank, and L is the order of the cepstrum.

$$c_n = \sqrt{\frac{2}{K}} \sum_{k=1}^{K} (log\ S_k)\ cos[n(k-0.5)\pi/K] \quad n = 1, 2, \ldots L$$

A cepstrum is a spectrum of a spectrum (in this case a DCT of an FFT) and it represents the periodicity of frequency contents. The resulting cepstrum is not in the time domain, nor the frequency domain, but rather the quefrency domain. Calculating the short time cepstrum of chunks of audio using a STFT results in a set of cepstrums that demonstrate how the periodicity of frequency content changes over time which is highly specific to individual sounds and is therefore very good for classification. In many ways, this method emulates how the human cochlea separates and identifies sounds based on frequency receptors.



2.3 Classification method

Given our data, a large set of normalized and labeled clips and predetermined categories, a neural network was clearly the best option for classification. The downsides to neural networks are very clear. The training data and its preparation is extremely important for getting results. This is especially true for audio where features like sampling rate and channel count can vary. Even within UrbanSound8K's data, normalization had to be applied before any classification could occur. For the purposes of this research, an artificial neural network (ANN) was used. Best-in-class algorithms often utilize convolutional neural networks (CNN) which add a level of sophistication that may increase accuracy but also increases computation time. We hypothesized that this would pose a problem when attempting real-time classification on the wearable, but further experimentation is required to test this theory.

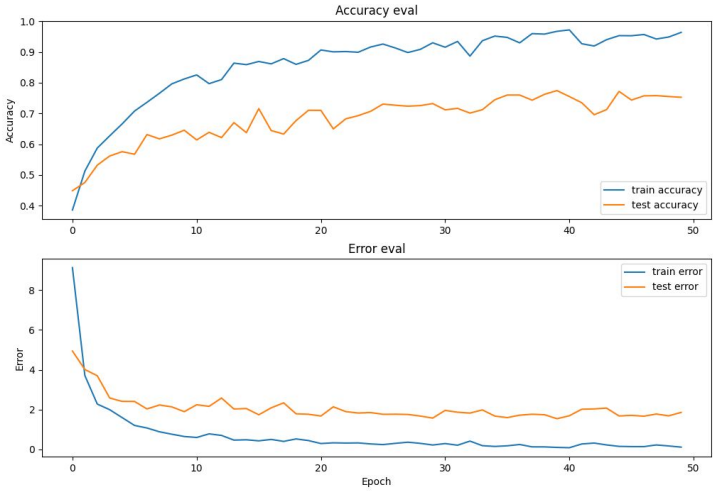
2.4 Training

The training process began by extracting MFCC features for all 8,732 audio clips in UrbanSound8K. The data was then loaded into a new file for building the deep learning model. Because of the way ANNs function, all of the MFCCs had to be converted into a one dimensional numpy array with a uniform shape. When the MFCCs are initially calculated they are in a 4-dimensional array of a varying quantity of sets of 13 MFCCs. The data must be flattened and then padded with zeros to a standardized length for the ANN to compute.

The ANN was built using Tensorflow, the most popular machine learning library.
The structure of the ANN is as follows:

Dense layer: 512 Neurons	Dense layer: 256 Neurons	Dense layer: 64 Neurons	Output layer: 10 Neurons

The number of neurons per layer gradually decreases as the data is filtered through until eventually reaching the ten output neurons corresponding to the ten classes.



3. Wearable

The wearable had to take outputs from the localization and classification algorithms we developed and display it to a user. As we imagine it being used to provide sound information to a person who is deaf or hard of hearing, we wanted our device to emulate the audio characteristics of human hearing. For the wearable, this meant placing it near the ears. Combined with the added benefit of providing a convenient location for the user display, a baseball cap became the chosen form factor.

3.1 Recording

Recording the four simultaneous audio streams in high definition required for localization is non-trivial in terms of processing requirements. Initial tests using a custom-made 4 input sound card resulted in an unusably slow sampling rate of less than 2000 Hz. We eventually settled on a low cost off-the-shelf microphone array designed to interface with the Raspberry Pi. The ReSpeaker contains 4 MEMS (Micro-Electro-Mechanical System) microphones arranged in a square configuration, spaced approximately 6 cm from each other. It provides 4 audio streams at 22,050 Hz which can be parsed and processed to determine a source's location.

3.2 Processing

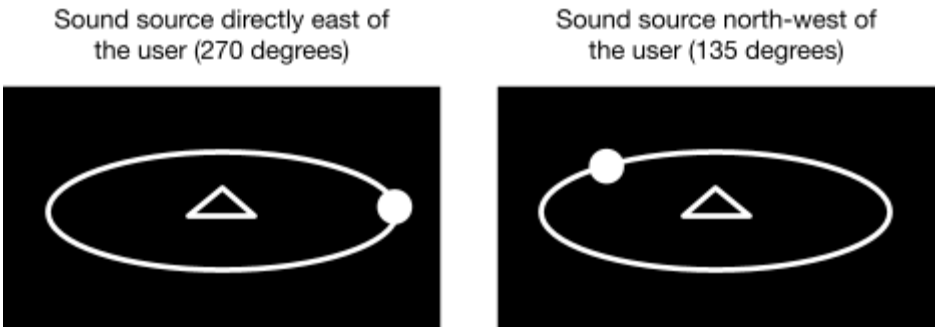
In order to provide real-time data to the wearer, the chosen processor needed to perform computations quickly while still fitting comfortably within a baseball cap. We settled on a Linux-based single-board computer (SBC), specifically, the Raspberry Pi 3A.

3.3 Inertial Measurement

An inertial measurement unit (IMU) keeps track of the wearable's rotational velocity. By integrating this data over time, we can compute the relative heading of the user. This means that when the user turns their head, the display updates instantly to show the correct relative sound source location. It allows the device to operate at more than 25 frame updates per second, making the user experience smooth and intuitive.

3.4 Display

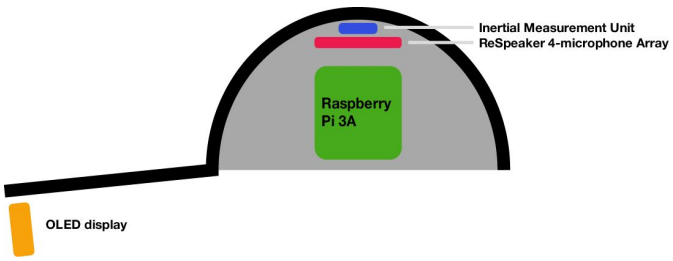
After the results of localization and recognition are computed by the Raspberry Pi, they must be displayed to the user through an easy to understand format. We chose a small off-the-shelf organic LED (OLED) display. A custom elliptical interface represents the user with a small arrow. A dot indicates the sound source location.



Sample interface views

After the individual components were selected we had to integrate them into a wearable device. After removing unused components from the printed circuit boards of the SBC and microphone array, they were embedded into the baseball hat. Silicone wire was used to connect the various components in order to maintain flexibility when worn.

3.5 Final Device



Hat component diagram

Completed hat with embedded SBC, microphone array and OLED display

OLED display while localizing a sound source

4. Integration

The individual components of the wearable, such as localization, machine learning, and hardware design, had been tested separately, but not in tandem. PyTest, an automated testing framework, proved invaluable in testing individual code and hardware.

4.1 Streaming Live Audio

All localization testing was performed on pre-recorded files to evaluate accuracy and speed. This was a significantly different scenario than the use-case of a real-time processing device. We abstracted the complexity of streaming live audio through object oriented programming.

Multithreading is a necessity because of the need for multiple parallel-like tasks. Multithreading only imitates parallelism, and in reality runs tasks concurrently. One thread reads a byte stream from the wearable containing all four audio channels. This data is then pushed onto an audio queue to avoid race conditions that can cause unpredictable behavior. The main thread of the program then processes the incoming audio queue to convert it to a format that is easily processed by other components. We deinterlace the audio channels from the sequential streams and update audio channel arrays to make them accessible through the class interface.

4.2 Different Window Size Requirements

Both machine learning and localization require different window sizes for the best results. Our localization algorithm works best with a buffer of 0.1 to 0.5 seconds of past signal, while the machine learning algorithm uses two to three seconds for optimal results. This would not have been a problem had the execution of the machine learning not depended on the execution of localization. Machine learning is executed once a minimum threshold of correlation has been established for an unknown sound that was received. A component from streaming audio was reused to constantly update different audio window sizes that resided in component specific classes.

4.4 Source Tracking

Reducing the executions of the machine learning algorithm was important to alleviate system resources. We made machine learning dependent on localization to increase the number of cycles of localization, while not sacrificing the benefit of sound recognition. By tracking recently identified sound sources, this significantly cutback on the number of ML calls. The main issue proved to be in providing the ML algorithm with source audio data. The sound received by ML was directly dependent on what sounds were ongoing, which resulted in subtle details and gaps in audio being removed.

Results

Each of the various algorithms and components of the final wearable were tested independently before being integrated into the final device.

Localization

The localization algorithm was tested during its development on test data created by a ReSpeaker and Raspberry Pi. This data was produced by recording each of the four signals from the microphone array into separate files while audio was played from a predefined angle.

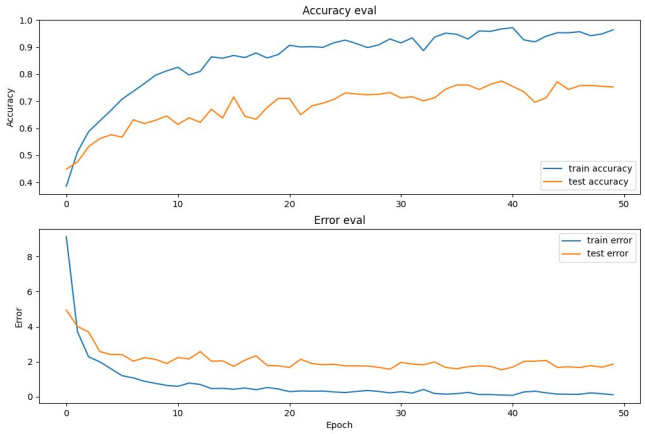
When it came time to test the validity of our localization algorithm, we played sounds with a smartphone at known angles and recorded the average predicted sound produced by our algorithm. The results were as follows:

Audio Type	Audio Source Angle (degrees)	Average Predicted Audio Source Angle (degrees)	Error
Dog Barking	360	356.6	0.94%
Drilling	45	41.8	7.11%
Street Music	180	183.7	2.06%
Siren	270	270.0	0%
		Average Error:	**2.53%**

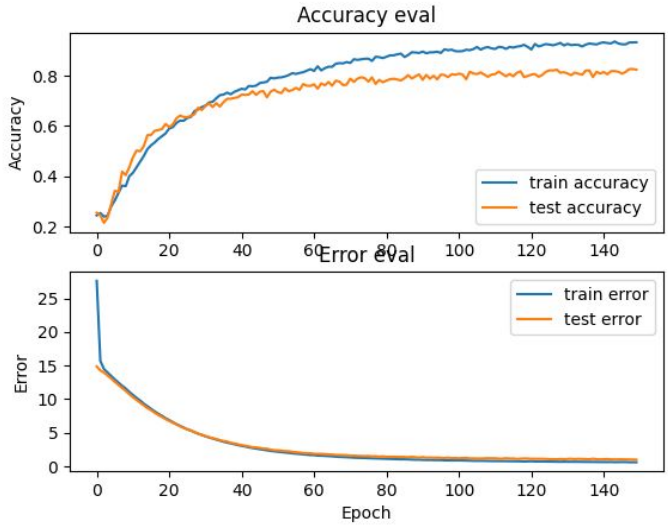
The localization algorithm shows accurate results at all angles around the device. Slightly higher error values are seen when the sound source is collinear with a pair of microphones, but is still accurate enough for user display.

Machine Learning

The machine learning model is tested using train-test split. This means that the data used to evaluate the accuracy of the final model is not data that it has already seen and should be able to evaluate. Naturally, the accuracy of test data will be lower than the accuracy on training data. When this difference is very large the algorithm is overfit, meaning it has learned to identify the input data too specifically and fails to apply its rules to new data. The difference between train and test accuracy and error can be seen in the graph below.



To solve this error, dropout layers were added in between each dense layer of the neural network. A dropout layer alters the neural network so that there is a 20% chance for a neuron on the previous layer to not be functional for that epoch. This forces the ANN to rely on all of its neurons rather than weighting a select few very heavily. On top of dropout, regularization is added to each layer which penalizes the model for creating large weights on individual neurons. After these tests were made, overfitting significantly decreased and the test accuracy increased by over 15%. The final model after training for 100 epochs has a train accuracy of 96.4% and a test accuracy of 84.6%.



Wearable Device

The wearable device was first tested on a stationary tripod where the localization test was repeated on the embedded device. Similarly performing results were achieved, indicating that the modified form-factor of the wearable did not affect the performance of our localization algorithm.

Next the wearable localization was repeated, this time while being worn by one of us. The display updated quickly as we changed our heading, using the IMU to interpolate source position in between localization calls.

Future Work

While our device performs well in localizing sound and displaying it to a user, and our machine learning algorithm is fairly accurate in recognizing its ten classified sounds, there are a number of improvements we would like to make and areas we would like to explore further.

The localization and classification machine learning algorithms should be unified on the embedded wearable to provide both positional and classification information to a user. We hope

that through this combination the device will be a useful tool for understanding the world through sound.

In order to reduce the effect of noise on our classification system, we hope to use steered beamforming to isolate the sound originating from a single location, after it has been identified by our localization algorithm. This would likely improve the recognition accuracy of our device.

A significant performance boost is expected in the use of multiprocessing versus multithreading due to a finer control of system resources. However communication between processes is more complicated compared to threading.

References

Gerhard, D. (2003, November). Audio Signal Classification: History and Current Techniques. Retrieved from http://www2.cs.uregina.ca/~gerhard/publications/TRdbg-Audio.pdf

Grondin, F. (2021, March 5). ODAS: Open embeddeD Audition System. Retrieved from https://arxiv.org/abs/2103.03954

Lyon, D. (2010, April). The Discrete Fourier Transform, Part 6: Cross-Correlation. Retrieved from http://www.jot.fm/issues/issue_2010_03/column2.pdf

Salamon, J. (2014, November). *A Dataset and Taxonomy for Urban Sound Research*. Orlando: 22nd ACM International Conference on Multimedia.

Turner, C. (2009, March 27). Time Reversal and Frequency Response. Retrieved from www.claysturner.com/dsp/timereversal.pdf