Detecting and Classifying Short-finned Pilot Whale Acoustics with Deep Learning

Virginia Pan

Abstract

Short-finned pilot whales are deep diving cetaceans that use sound for navigation, hunting, and communication. Scientists use digital acoustic recording tags (DTAGs) to collect depth, movement, and audio data that provide insight on their behavior. Typically, someone listens to recorded DTAG audio and annotates the start time, duration and type of acoustic event they hear. However, manual review of DTAG audio recordings can be time consuming and requires a trained ear. Thus, the goal of this project is to automate the processing of DTAG audio by developing a detection and classification scheme. This would give scientists more time to dive into looking for trends in the data rather than processing it, which is currently very time and labor intensive. This thesis is composed of two parts: classification and detection. In Part I, I differentiated between short-finned pilot whale buzzes and minibuzzes using deep learning. In Part II, I used a time domain detection scheme to find pilot whale vocalizations in the audio recordings that could be extracted and classified using the neural network. Lastly, I summarize the progress towards he ultimate goal of combining the detection and classification steps for a complete workflow to enable scientists to more quickly analyze DTAG audio without having to aurally review entire records.

Introduction

Globicephala macrorhynchus, commonly known as short-finned pilot whales, are a species of odontocete globally distributed around the world's tropical and temperate oceans [1]. Like other cetaceans, they are a long lived and slow to reproduce species that live in highly social stable family groups of 15 to 30 individuals [1]. They use echolocation to hunt for fish and squid at depths exceeding 1,000 feet [1]. They are called "cheetahs of the deep sea," due to their deep, high speed dives to chase and capture large squid [1].

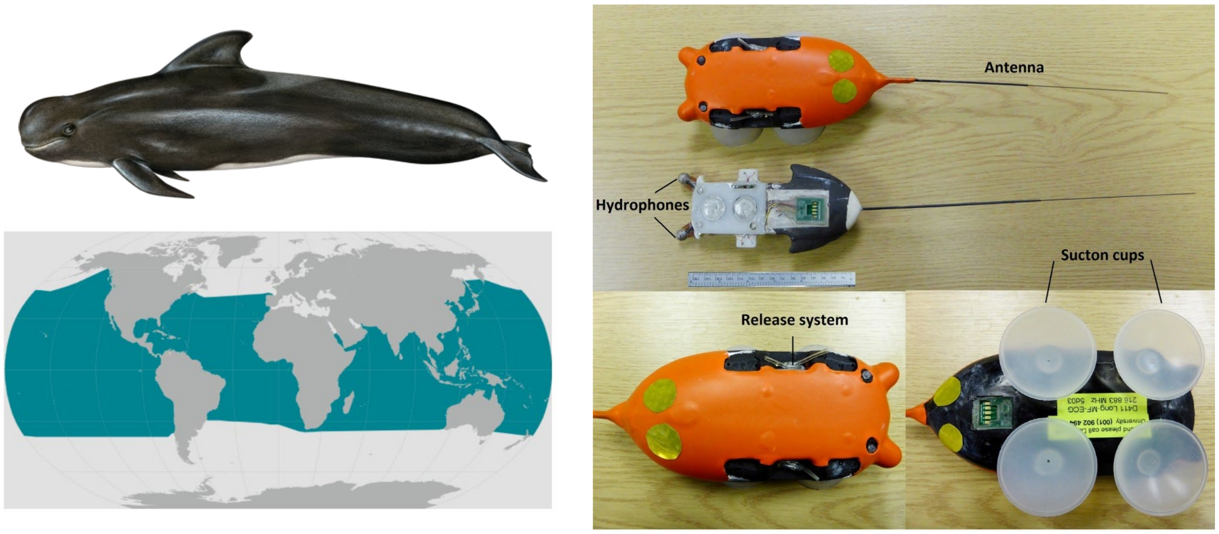


Figure 1. Background: Short-finned pilot whale (top left) [1], Short-finned pilot whale distribution (bottom left) [1], DTAG diagrams (right) [2]

Digital Acoustic Recording Tag (DTAG) is a suction cup tag used to study short-finned pilot whales and other cetaceans. The DTAG records audio, pitch, roll, heading and depth of the animal [2]. Together, this data allows scientists to gain insight on short-finned pilot whale social interactions, foraging, diving and ecology. Researchers from the Duke Marine Lab traveled offshore Cape Hatteras, NC and tagged 25 individuals total in 2008, 2010, and 2011.

Short-finned pilot whales make many different types of noises, including buzzes, minibuzzes, calls and clicks. Clicks are produced during echolocation to navigate and search for prey. Buzzes and minibuzzes are used towards the end of prey capture to hone in on their target. Calls are used in social interactions and are very variable.

Literature Review

In audio detection and classification literature, the two primary approaches are to perform detection and classification at the same time or separately.

I reviewed several articles on audio detection and classification in order to understand current methodologies. Generally, articles I reviewed for automatic detection and classification fell into two main groups: classical techniques for marine mammal passive acoustic monitoring (PAM) systems and learning-based approaches for other animal sounds.

In *Automatic Detection and Classification of Odontocete Whistles,* Gillespie et al. focused on contour analysis in order to analyze whistles and made algorithms robust to fragmentation of whistles, however they did not look at clicks or buzzes [3]. Lin similarly used a contour analysis approach for whistles and characterized buzzes from their fundamental frequency and harmonics [4]. An article by Bittle and Duncan provided an overview of marine mammal PAM techniques as of 2013, including detection and feature extraction, energy-based algorithms, Hidden Markov models, Gaussian mixture models, neural networks, information entropy, and Hilbert-Huang transform [5].

In contrast, learning-based approaches use an optimization process to automatically generate parts of the algorithm. Deep learning is a subset of machine learning, which is a subset of artificial intelligence. In machine learning, a human designs features for the program to pay attention to, which determines what class the input belongs to. However, in deep learning, the features are learned and thus one does not need to know what features are important to differentiate between the classes. The tradeoff is that large amounts of data input are needed. This is ideal for my application, because I do not already know what the key differences are between buzzes and minibuzzes are. Deep learning typically requires a high-performance GPU, lots of examples and training time, but can yield highly accurate results with minimal domain-specific expertise.

Interesting work using deep learning for automatic detection and classification of acoustics in other species, such as lab mice, elephants, and monkeys shows promise for DTAG applications [6, 7, 8]. Bjorck et al. used a convolution-LSTM (long short-term memory) hybrid network on fixed length audio segments for real time detection, compression, and transfer of elephant audio for further analysis [6]. DeepSqueak used regional convolutional neural network architecture (Faster-RCNN) for detection and k-means for classification focused on contour characterization for analysis of lab mice squeaks [7]. However, mouse trills (analogous to buzzes) appeared difficult to isolate via contour-based approaches [8]. Perhaps most promising, Oikarinen et al. used an end-to-end feed forward CNN on fixed length recordings snippets for detection, classification, and attribution of captive marmoset monkey sounds [8].

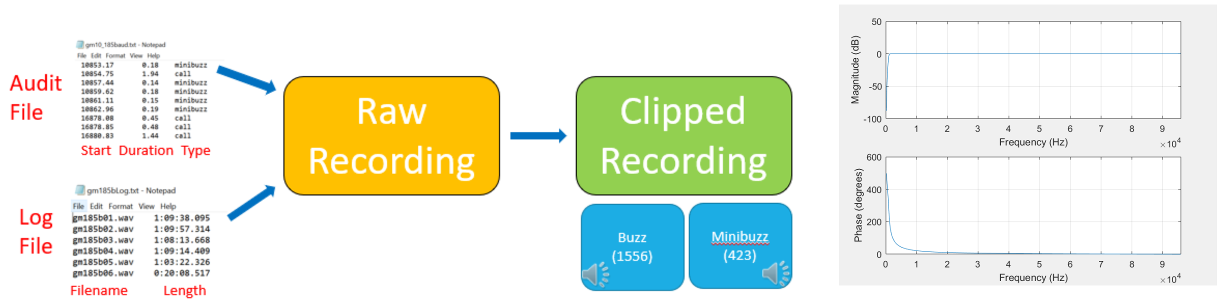
Although learning-based approaches for combined detection and classification show good results, they lack milestones to assess progress and fail to provide modularity. The result is a monolithic algorithm with limited ability to adapt. Therefore, I chose to analyze classification and detection separately. By doing so, I can experimentally validate each step separately.

The primary methods considered for classification include machine learning on hand-designed features, deep learning using time-domain waveforms, and deep learning using frequency-domain waveforms. Because I lack domain-specific expertise on pilot whale acoustic waveforms, hand-designed features would be difficult to invent. Literature studying deep learning approaches using time-domain input for classification show good accuracy at the expense of increased memory and computational requirements [9]. Another approach is to use deep learning on frequency domain data. In frequency-domain approaches, methods from image classification can be leveraged by representing the Short Time Fourier Transform (STFT) as an image. The challenge that both time domain and frequency domain approaches face is dealing with variable-length audio data. [10] deals with this problem using max-pooling.

PART I: CLASSIFICATION

Methods

I choose to use deep learning for classifying buzz and minibuzz vocalizations. I choose to focus on buzz and minibuzz sounds because they are contained, consistent, and are of biological interest, because they are used during the final stages of feeding. This is opposed to clicks, which happen over a longer period and calls, which are complex and varied in structure. I started working with data from 2010, which came from 11 separately tagged individuals and resulted in 56 hours and 57 minutes of DTAG audio. Later, I incorporated recordings from 5 individuals tagged in 2008 and 9 individuals from 2011. The DTAG stores hydrophone audio in approximately one-hour chunks. A log file was made for each individual, which listed the audio filename and precise length of each audio chunk from an individual in chronic order. Audit files were manually prepared for each individual pilot whale, that noted the start time, duration, and vocalization type. Information from the audit file and log file were used to extract the clips of audio that contained buzzes and minibuzzes. The audio splitting was preformed in Python with the Pydub library.


```
gm10_185baud.txt - Notepad
File  Edit  Format  View  Help
 10853.17       0.18    minibuzz
 10854.75       1.94    call
 10857.44       0.14    minibuzz
 10859.62       0.18    minibuzz
 10861.11       0.15    minibuzz
 10862.96       0.19    minibuzz
 16878.08       0.45    call
 16878.85       0.48    call
 16880.83       1.44    call
```

```
gm185bLog.txt - Notepad
File  Edit  Format  View  Help
gm185b01.wav    1:09:38.095
gm185b02.wav    1:09:57.314
gm185b03.wav    1:08:13.668
gm185b04.wav    1:09:14.409
gm185b05.wav    1:03:22.326
gm185b06.wav    0:20:08.517
```

Figure 2. Audio Pre-processing: Workflow for extracting buzz and minibuzz audio segments from DTAG recordings (left), Characterization of magnitude and phase response of the high pass Butterworth filter with cutoff frequency of 1kHz (right)

In general, minibuzzes tended to be shorter (most were less than half a second), while buzzes tended to be longer (most were between half a second and three seconds), with the longest buzz being 19 seconds. All the longer recordings were kept, however some of the recordings that were too short to contain useful information were thrown out.

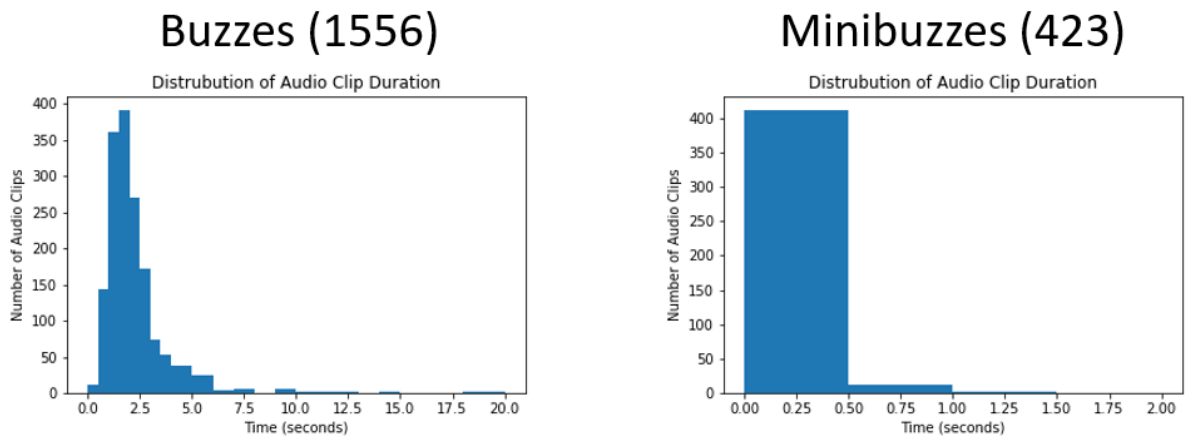


Figure 3. Length distribution of buzzes and minibuzzes for all 2010 recordings

Next, a high pass Butterworth filter with a cutoff frequency of 1kHz was designed and applied in Matlab to eliminate low frequency noise. A short time Fourier transform (STFT) was preformed on the filtered audio clips. STFTs were performed using the Librosa library [11] in Python with a N of 2048, window length of 1024 and step size of 512. These parameters were determined after experimentation to find the best-looking spectrogram for selected audio clips (trade off between time and frequency resolution). The STFT served as the input for the deep learning network.

1,556 buzzes and 423 minibuzzes were extracted from all the 2010 recordings. All minibuzzes were used and 423 of the 1,550 buzzes were randomly selected to be used to train and test the model. 65% were used for training, 15% were used for validation and 20% were used for testing.

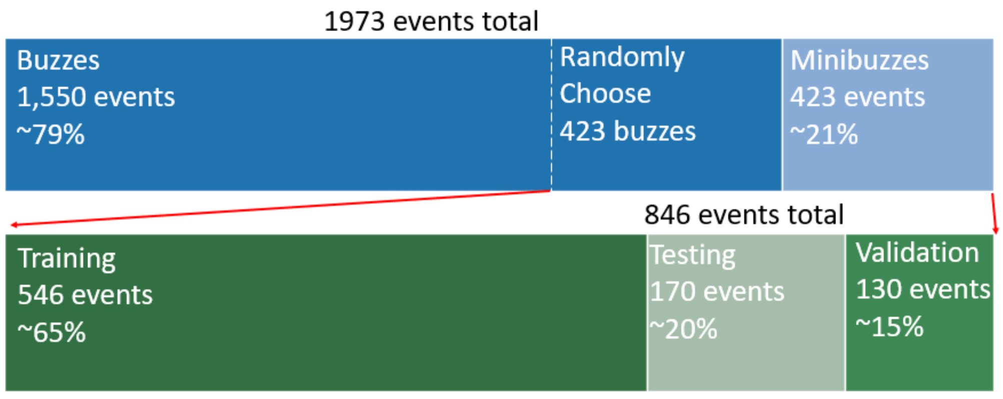


Figure 4. Breakdown of all buzzes and minibuzzes from 2010 recordings, and how they were distributed amongst training, validation and testing

The basis for the neural network I used was outlined in "Classifying Variable-Length Audio Files with All-Convolutional Networks and Masked Global Pooling" [10]. The input to the network were STFTs of the filtered buzz and minibuzz clips, with the correct classification labels from the audit file. The output was classification of the clip as a buzz or minibuzz. One of the primary challenges of working with this dataset was having different length audio segments. This was resolved by using a global pooling layer right before the fully connected layer.

The input STFT was first convolved with 256, 1 by 1025 filters to identify the presence, of 256 unique sounds. Then a reLU activation layer was applied before convolution with 256, 3 by 256 filters with an overlap of 2. The resultant array was approximately half as long in the time dimension and was put through another reLu layer. This combination of convolution and reLu layers was repeated for a total of three times. Functionally, this grouped sounds into sequences. Next, the global max pooling layer was applied to each column, to check for the presence of a sound sequence in the whole audio clip. Finally, there was a fully connected layer, in which each cell from the max pooled layer "cast a vote" to whether the clip was a buzz or minibuzz.

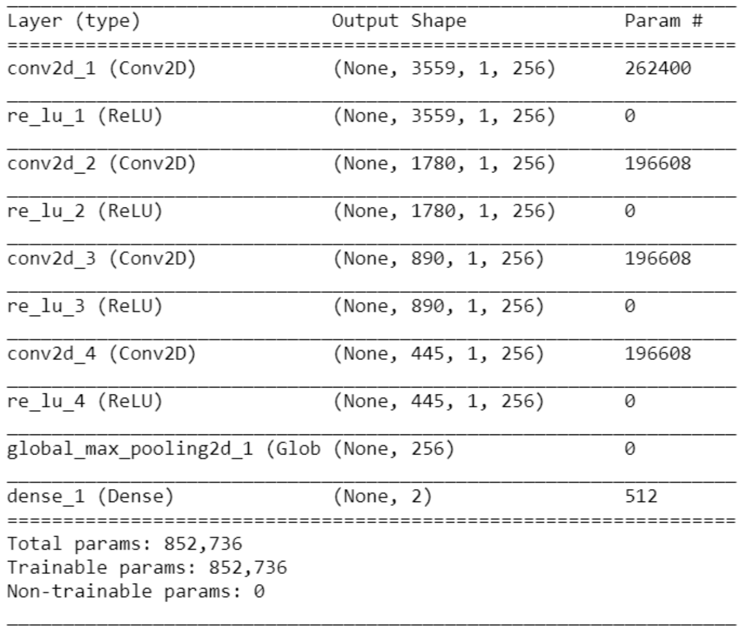

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 3559, 1, 256)      262400
_________________________________________________________________
re_lu_1 (ReLU)               (None, 3559, 1, 256)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 1780, 1, 256)      196608
_________________________________________________________________
re_lu_2 (ReLU)               (None, 1780, 1, 256)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 890, 1, 256)       196608
_________________________________________________________________
re_lu_3 (ReLU)               (None, 890, 1, 256)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 445, 1, 256)       196608
_________________________________________________________________
re_lu_4 (ReLU)               (None, 445, 1, 256)       0
_________________________________________________________________
global_max_pooling2d_1 (Glob (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 512
=================================================================
Total params: 852,736
Trainable params: 852,736
Non-trainable params: 0
_________________________________________________________________
```

Figure 5. Convolutional Neural Network Model Summary

The deep learning was preformed using Amazon Web Services (AWS). The STFT of each clip was uploaded and stored on the cloud in the AWS S3 storage system. Data was brought into my Amazon Sagemaker Jupyter Notebook workspace from S3. I coded the model in Python 3 with Keras backed by Tesnsorflow.

Results

I first trained the model with an equal distribution of buzz and minibuzz events from 2010 (676 events total) and tested it with an equal distribution 170 new of buzz and minibuzz events from 2010. This resulted in a 97.633% accuracy. I then went on to test the system with all 230 events (95 buzz, 135 minibuzz) from 5 short-finned pilot whales tagged in 2008. This resulted in a 90.533% accuracy. Next, I tested the model with 575 events from 2011 (483 buzzes and 92 minibuzzes). The model preformed quite poorly on this data, with an accuracy of only 53.254%.

Unsatisfied with these poor results, especially for the 2011 data, I decided to retrain and test the model with buzzes and minibuzzes from all three years. I trained it with 532 examples, including an equal number of buzzes and minibuzzes from across all three years. I tested it with 768 examples, which also included an equal number of buzzes and minibuzzes. This resulted in a 97.633% accuracy.

Discussion

Overall, I am satisfied with the accuracy of the model and realize that the robustness could be further improved by incorporating more data from different individuals. I wonder if the model preformed so poorly on the 2011 dataset because there was way more buzzes than minibuzzes (84% buzzes, 16% buzzes) as compared to the evenly split dataset the model was trained on.

One of the biggest questions I faced in this process was deciding the distribution of buzzes and minibuzzes to train the model with. I decided to train with an equal number of buzzes and minibuzzes to prevent the model for preferentially predicting one or the other. For example, in an extreme case if a model is trained with 97% class A examples and 3% class B examples, it may learn to classify all input as class A, which would make it 97% correct, but fail to recognize any class Be events, which maybe more important.

After looking at the results, I am curious if I should have trained the model with a breakdown of minibuzzes and minibuzzes proportional to the real-life. In the 2010, 2008, and mixed years test, the breakdown between buzzes and minibuzzes was roughly equal, while in 2011 there were way more buzzes than minibuzzes. To investigate this, I could see if the model incorrectly classified many buzzes as minibuzzes and correctly classified all the minibuzzes.

PART II: DETECTION

Methods

I used the first clip from individual "gm10_188a" as my test file, because it was relatively shorter (around 35 minutes) compared to most hour-long segments and had an abundance of sound events. The whole audio file was filtered using the same 1kHz high pass Butterworth filter as used in the classification scheme.

I chose to try detection using the auditok library [12] because of its flexibility for parameter tuning and up-to-date documentation. I ran the detection with a variety of energy threshold values, ranging from 30 to 50 in increments of 2, where energy is calculated as:

$$energy = 10 * \log_{10} \frac{\|signal\|_2^2}{\text{length}(signal)}$$

All other parameters, including minimum signal length, maximum signal length, and maximum length of silence period within signal, were kept constant at 0.1s, 20s, and 0.5s respectively.

After running detection, detected regions were compared to the audit file to evaluate precision, recall, and accuracy. Metrics are calculated as:

$$precision = \frac{total\ overlap\ time}{total\ detected\ time}\quad .$$

$$recall = \frac{total\ overlap\ time}{total\ audit\ time}\quad .$$

$$accuracy = \frac{(total\ overlap\ time)+(total\ nonoverlap\ time)}{total\ time}$$

$$= 1 - \frac{(total\ audit)+(total\ detected)-2*(total\ overlap)}{(total\ time)}$$

The results with waveforms were also graphed for visual inspection.

Results

Detection results were output to a text file with event number, start time, and end time. Detected events are strictly non-overlapping. Results were plotted, including waveforms, detection regions, audit regions, and overlap of detection and audit regions. Selected results for energy threshold values of 30, 40, and 50 are shown in Figures 7-9.

Graphing the waveform posed difficulties due to high sampling rate and duration of audio file. Several techniques were required to make both graphing and analyzing the audio file computationally tractable. First, sections of the audio file were read in chunks of 5 million samples (~26s) to limit the amount of data loaded into RAM. However, the plotting library had difficulty displaying the large number of data points (192000 points per second). Initially, I tried to naively down sample the audio by plotting every thousandth point, however this significantly altered the visual shape of the waveform. Qualitatively, it was significantly smaller and less dense. Next, I tried plotting the envelope as extracted using a Hilbert filter. This was better in that it captured the general shape of the waveform more accurately, however it lacked in capturing the dynamic range of the audio waveform. Finally, I used the librosa library [11] for plotting the waveform, which allowed me to specify the number of data points to plot for each segment of audio. Libro sa fills a curve specified by $[-|y|, |y|]$ where y is the largest amplitude in the audio window with length $\frac{T}{n}$ where T is the length of the original audio segment (~26s) and n is the number of points to plot (1000). This corresponds to down sampling the audio to a 38.46Hz sampling rate using the envelope. Figure 6 shows the original waveform, naïve down sampling, Hilbert filter, and librosa plots respectively on a test audio file.

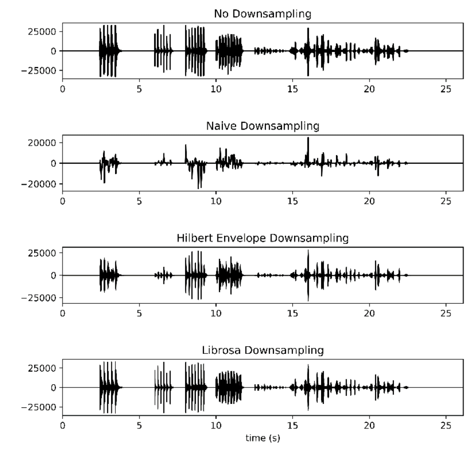


Figure 6: Comparison of Waveform Plotting

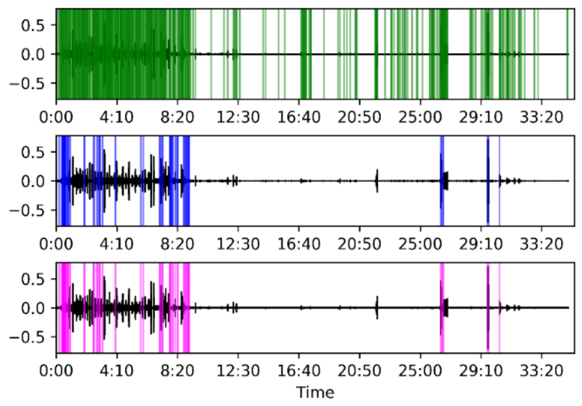


Figure 7: Energy threshold of 30: Detection regions (top), audit detections (middle), overlap of detections and audits (bottom)

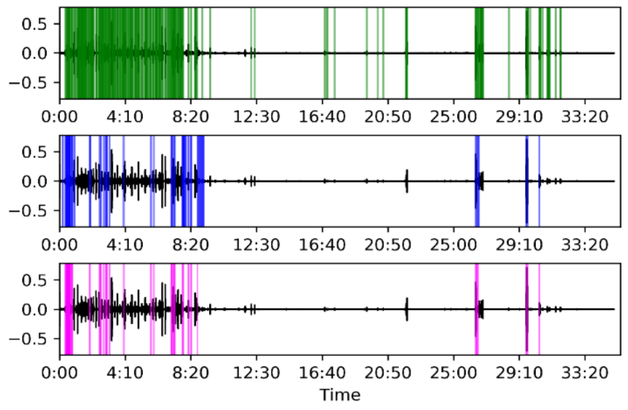


Figure 8: Energy threshold of 40: Detection regions (top), audit detections (middle), overlap of detections and audits (bottom)

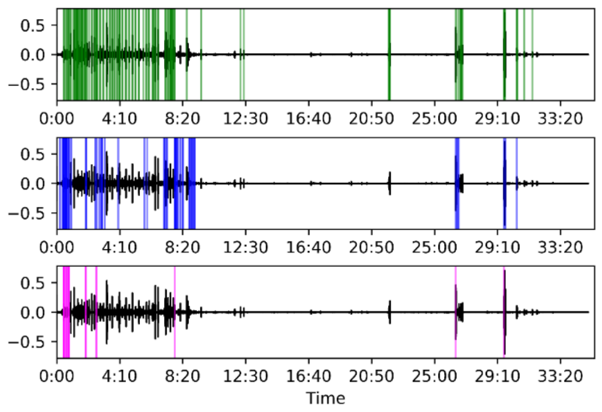


Figure 9: Energy threshold of 50: Detection regions (top), audit detections (middle), overlap of detections and audits (bottom)

Precision, recall, and accuracy were calculated for each energy threshold value and shown in Figure 10 and Table 1.

Discussion

Lower energy threshold values captured more audit events, increasing recall without noticeably decreasing precision. The fact that precision does not appreciably change with energy threshold evidences that more actual audit sounds were detected by decreasing energy threshold. Furthermore, since precision represents the ratio of correctly identified detections to total detections, auditok exhibits an overall increase in the number of correctly detected events proportional to the number of total detected events as we lower the energy threshold.

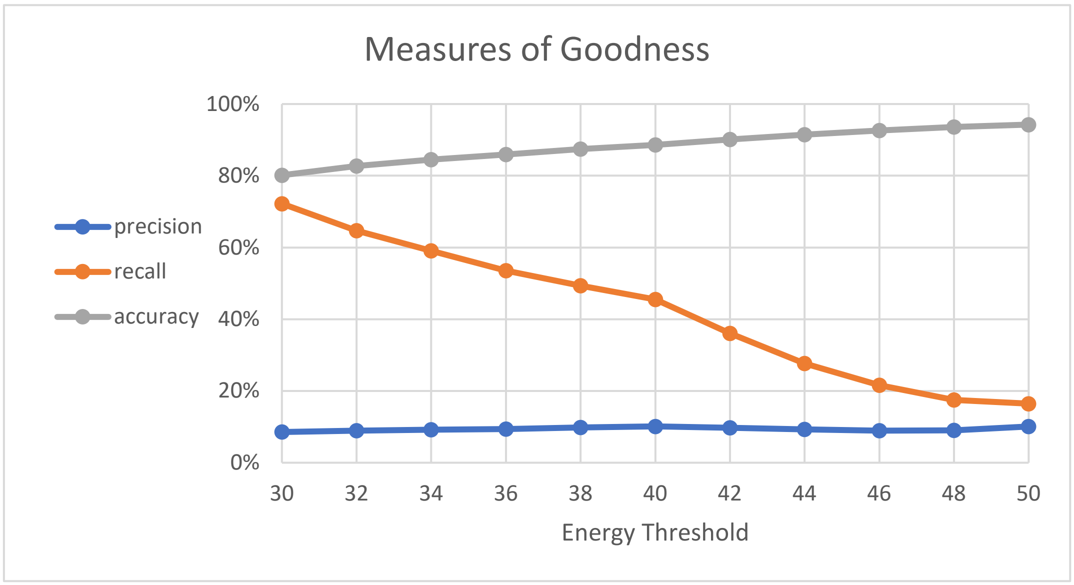


Figure 10: Detection performance metrics

Table 1: Detection Performance Metric Data

Energy Threshold	Precision	Recall	Accuracy
30	9%	72%	80%
32	9%	65%	83%
34	9%	59%	85%
36	9%	54%	86%
38	10%	49%	87%
40	10%	45%	89%
42	10%	36%	90%
44	9%	28%	91%
46	9%	22%	93%
48	9%	18%	94%
50	10%	16%	94%

Accuracy, however, drops as energy threshold decreases because more non-audit sounds were detected. Accuracy represents both true positives and true negatives. The number of true positives increases as energy threshold decreases, but the increase in the number of false positives causes a decrease in the true negatives which in turn causes the overall accuracy to drop.

We primarily want to maximize recall so that we do not miss audit events. The classification network can be trained to identify flow noise captured in false positives, which means that it can compensate for the low precision. Accuracy roughly corresponds to how difficult it will be to train the network due to the large proportion of flow noise as compared to real audit events. Therefore, the current results suggest that a lower energy threshold will be beneficial for our detector/classifier scheme.

The detector was successful at several aspects. First, it ignored the silent periods. As we decreased the energy threshold, the locations of additional detections were clustered around true audit regions and regions of the waveform which were qualitatively more interesting. Additionally, the recall does attain reasonable recall rates at 72% for low energy threshold values. It also demonstrates a clear relationship to the parameters which indicates that further parameter tuning has the potential to improve performance further.

However, the detector also has many shortcomings. Fundamentally, the detector acted as a peak-detector while many of the audit events do not occur at peak sound levels. In addition, the detector breaks up continuous sound that should be processed together as a single event. This may be mitigated by increasing the "maximum length of silence within signal" parameter. The detector also does not allow for overlapping events, which are present in the audit file and real life.

Given these many shortcomings, this classical detection method will not be further pursued in favor of deep learning methods with simultaneous detection and classification.

SUMMARY

Future Work

If one were to continue developing classical detection schemes, I would recommend analysis in the frequency domain in order to create more space for feature extraction. A combination of frequency and time domain parameters may produce sufficient detection performance. Alternatively, extensions upon naïve pattern matching, template matching, or nearest neighbor algorithms in the frequency domain could be used but would require different "templates" for each sound event type, would not be robust to varying length sounds, and might not work for atypical sounds.

Several detection techniques have also been developed recently using deep learning techniques. Existing literature on sound detection and classification in other animals typically chunks the audio into segments for STFT neural network processing. One promising method [8] is to use a purely feed-forward CNN on a continuously sliding window of the most recent audio spectrogram data. This also has additional appeal in our project because the CNN developed last semester for classification can be reused with minimal modification by continuously classifying sound chunks using a sliding window. Furthermore, the CNN developed last semester also supports some duration invariance which could improve upon the results attained by [8]. This method is similar to the R-CNN (and later fast R-CNN and faster R-CNN) methods commonly used in image detection and in [7].

Furthermore, the model could be improved by training it with more data from more individuals and trained to classify other types of acoustic events, such as calls. Ultimately, from a scientific perspective, there are tons of relationships to explore between the acoustic events and other DTAG data (depth and movement) to gain insight on short-finned pilot whale foraging behavior.

Acknowledgements

I would like to thank the Nowacek Acoustics and Engineering Lab and the Read Lab for gathering the short-finned pilot whale DTAG data. Thank you to Dr. Doug Nowacek and Dr. Nicola Quick for providing me with the DTAG data, information about short-finned pilot whales and methodology guidance. Lastly, thank you to the Pratt School of Engineering, Duke University Marine Lab, and Rachel Carson Scholars Program.

References

[1] NOAA. “Short-Finned Pilot Whale,” Retrieved May 4, 2019, from https://www.fisheries.noaa.gov/species/short-finned-pilot-whale

[2] M. P. Johnson and P. L. Tyack, "A digital acoustic recording tag for measuring the response of wild marine mammals to sound," in *IEEE Journal of Oceanic Engineering*, vol. 28, no. 1, pp. 3-12, Jan. 2003.

[3] D. Gillespie, M. Caillat, J. Gordon, and P. White, “Automatic detection and classification of odontocete whistles,” *The Journal of the Acoustical Society of America*, vol. 134, no. 3, pp. 2427–2437, 2013.

[4] T.-H. Lin, H.-Y. Yu, C.-F. Chen, and L.-S. Chou, “Automatic detection and classification of cetacean tonal sounds from a long-term marine observatory,” *2013 IEEE International Underwater Technology Symposium (UT)*, 2013.

[5] M. Bittle and A. Duncan, “A review of current marine mammal detection and classification algorithms for use in automated passive acoustic monitoring,” *Proceedings of Acoustics*, vol. 2013, Nov. 2013.

[6] J. Bjorck, B. H. Rappazzo, D. Chen, R. Bernstein, P. H. Wrege, and C. P. Gomes, “Automatic Detection and Compression for Passive Acoustic Monitoring of the African Forest Elephant,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 476–484, 2019.

[7] K. R. Coffey, R. G. Marx, and J. F. Neumaier, “DeepSqueak: a deep learning-based system for detection and analysis of ultrasonic vocalizations,” *Neuropsychopharmacology*, vol. 44, no. 5, pp. 859–868, Apr. 2019.

[8] T. Oikarinen, K. Srinivasan, O. Meisner, J. B. Hyman, S. Parmar, R. Desimone, R. Landman, and G. Feng, “Deep Convolutional Network for Animal Sound Classification and Source Attribution using Dual Audio Recordings,” *bioRxiv*, Jul. 2018.

[9] Z. Zhu, J. H. Engel, and A. Hannun, “Learning multiscale features directly from waveforms,” 2016.

[10] Hertel, Lars & Phan, Huy & Mertins, Alfred, “Classifying Variable-Length Audio Files with All-Convolutional Networks and Masked Global Pooling,” 2016

[11] B. Mcfee, C. Raffel, D. Liang, D. Ellis, M. Mcvicar, E. Battenberg, and O. Nieto, “librosa: Audio and Music Signal Analysis in Python,” *Proceedings of the 14th Python in Science Conference*, 2015.

[12] https://github.com/amsehili/auditok