SEPTEMBER 29 2023

Machine learning methods for finding the best wavelet for audio compression

Shaun Pies ; Avery Landeche; Kendal M. Leftwich ; Juliette Ioup

Proc. Mtgs. Acoust. 51, 055002 (2023)
https://doi.org/10.1121/2.0001782

Volume 51 http://acousticalsociety.org/

184th Meeting of the Acoustical Society of America

Chicago, Illinois

8-12 May 2023

Signal Processing in Acoustics: Paper 2aSP8

Machine learning methods for finding the best wavelet for audio compression

Shaun Pies, Avery Landeche, Kendal M. Leftwich and Juliette Ioup
Department of Physics, University of New Orleans, New Orleans, LA, 70148; spies@uno.edu; aclandec@uno.edu; kmleftwi@uno.edu; jioup@uno.edu

Audio compression is vital in various applications, enabling efficient storage and transmission of audio signals while preserving perceptual quality. Wavelets can be remarkably effective at compressing data, but the choice of an appropriate wavelet is crucial for achieving optimal compression performance. In this paper, we show the use of machine learning techniques to automatically identify the most suitable wavelet for audio compression. Specifically, we use the Python programming language and the Scikit-Learn library along with the cross-correlation mathematical technique to choose the best wavelet based on the similarities between the compressed signal and the original signal.

Published by the Acoustical Society of America

1. INTRODUCTION

Audio compression is essential in today's digital world. With the proliferation of digital media and cloud storage for data, compressing files to reduce storage requirements and bandwidth for transmission of data is of paramount importance. One method for compressing audio signals is the use of wavelets. Wavelets have the ability to reduce file size by 50% using only one level of decomposition. However, using wavelets is not a straightforward process. There are many different wavelets to choose from and each one will have a different outcome when applied to a specific dataset. Traditionally, wavelet selection has been a manual process relying on expertise and experience with wavelet selection as well as knowledge of the data. In this paper we explore a novel method of choosing wavelets for audio compression by employing machine learning techniques.

We start by first defining what the best wavelet for compression means by use of the cross correlation. We use the python programming language and the Pywavelets[1] library to decompose audio signals and reconstruct them with zero details. The cross correlation was calculated to evaluate the similarity between the original data and the reconstructed data. The wavelet decomposition that had the highest cross correlation value was deemed the best wavelet. This data was then input into a decision tree classifier to train a machine learning model. Then the model was tested with data that was not used for training the model and the results were compared to the results of the cross correlation method for choosing a wavelet.

We show that machine learning can be an effective tool for choosing a wavelet to compress audio signals. Once the model has been trained the selection process is very accurate and efficient. During this process we also discovered that for most audio data that we tested, the discrete Meyer wavelet proved to be the best wavelet for compression.

2. WAVELET ANALYSIS

Wavelets are defined as wave-like basis functions $w_{a,s}(t)$, with the amplitude starting from zero, followed by increases or decreases, and terminating at a value of zero.[2] These basis functions are continuous in time and are described by Eq. 1.

$$w_{a,s}(t) = |a|^{-1/2} w(\frac{t-s}{a}) \tag{1}$$

where a is the scale factor and s is the shift in time t.

The continuous wavelet transform (CWT) of a function of time $f(t)$ is given by Eq. 2, and the inverse continuous wavelet transform is given by Eq. 3.

$$F_w(a,s) = \int_{-\infty}^{\infty} f(t) w_{a,s}(t) dt \tag{2}$$

$$f(t) = \frac{1}{C} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} a^{-2} F_w(a,s) w_{a,s}(t) \, da \, ds \tag{3}$$

The scale a replaces the frequency ω in the wavelet transform. The constant C is given by: $C \equiv 2\pi f |\widehat{w}|^2 \frac{d\omega}{|\omega|}$, where $\widehat{w}(t)$ is referred to as the mother wavelet. Smith[3] defines the discrete wavelet transform (DWT) as Eq. 4, and its inverse is defined by Eq. 5.

$$F_w(k,n) = s^{-\frac{k}{2}} \int_{-\infty}^{\infty} f(t) w(na^k T - t) dt \tag{4}$$

$$f(t) = \sum_k \sum_n F_W(k,n)\varphi_{kn}(t) \tag{5}$$

The mother wavelet w can be interpreted as a continuous impulse response filter. The basis of the wavelet function is denoted as φ.

Filter banks provide an alternative understanding of the discrete wavelet transform (DWT). The Python library Pywavelets,[1] for instance, utilizes filter banks for performing the DWT. The basic filter bank consists of a combination of high-pass filters $h_H(n)$ Eq. 6 and low-pass filters $h_L(n)$ Eq. 7.

$$y_{high}(n) = \sum_{k=-\infty}^{\infty} h_H(k)x(n-k) \tag{6}$$

$$y_{low}(n) = \sum_{k=-\infty}^{\infty} h_L(k)x(n-k) \tag{7}$$

One level of decomposition is illustrated in the following block diagram:

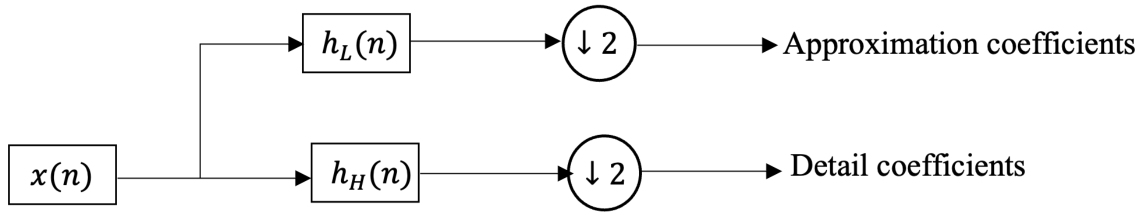

Figure 1: Block diagram showing one level of wavelet decomposition

The down-sampling operator "down 2," denoted as (↓2), is defined as follows: $(\downarrow 2)h(n) = h_{even}(n) = \{h(0), h(2), ...\}$. A low-pass filter is represented by a moving average and has the properties: $h_L(k) = \left\{\frac{h(0)+h(1)}{2}, \frac{h(1)+h(2)}{2}, \frac{h(2)+h(3)}{2}, \ldots, \frac{h(N-1)+h(N)}{2}\right\}$ for a causal Finite Impulse Response (FIR) low-pass filter, where the impulse response satisfies $\sum h_L(k) = 1$. On the other hand, a high-pass filter calculates differences between data points and has the properties:

$h_H(k) = \left\{\frac{h(0)-h(1)}{2}, \frac{h(1)-h(2)}{2}, \frac{h(2)-h(3)}{2}, \ldots, \frac{h(N-1)-h(N)}{2}\right\}$ for a causal FIR high-pass filter, where $\sum h_L(k) = 0$. The approximation coefficients are obtained from the low-pass filter and represent smoothed or averaged data coefficients after down-sampling. The detail coefficients indicate the differences or changes between the data points for the down-sampled data.

Equations 6 and 7 can be viewed as convolutions between the transmitted signal and the low-pass or high-pass filters. A block diagram illustrating multilevel decomposition is showin in Figure 1.

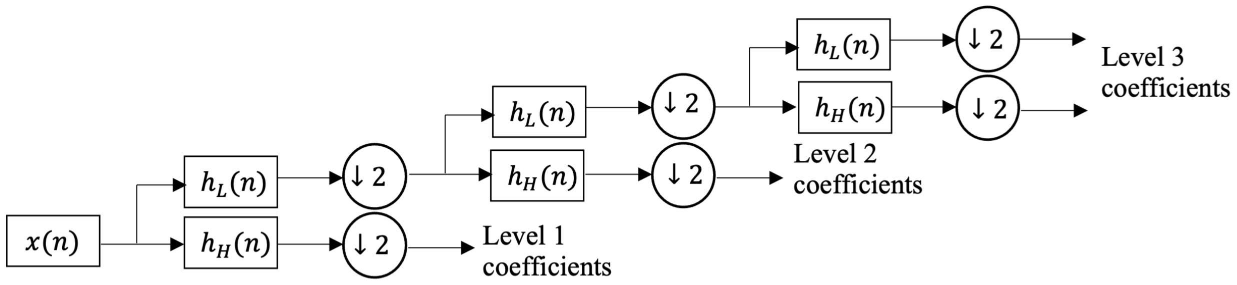

Figure 2: Block diagram showing 3 levels of decomposition

At each level of decomposition in the DWT, the approximation coefficients become smoother, while the detail coefficients capture the differences between the down-sampled values. It is crucial to be aware of these characteristics during the DWT process since data loss occurs with each decomposition level. The objective is to utilize this decomposition to eliminate unwanted data points. Figure 3 presents the approximation and detail coefficients of a level 4 decomposition of a sperm whale signal using the Daubechies 6 (db6) wavelet.

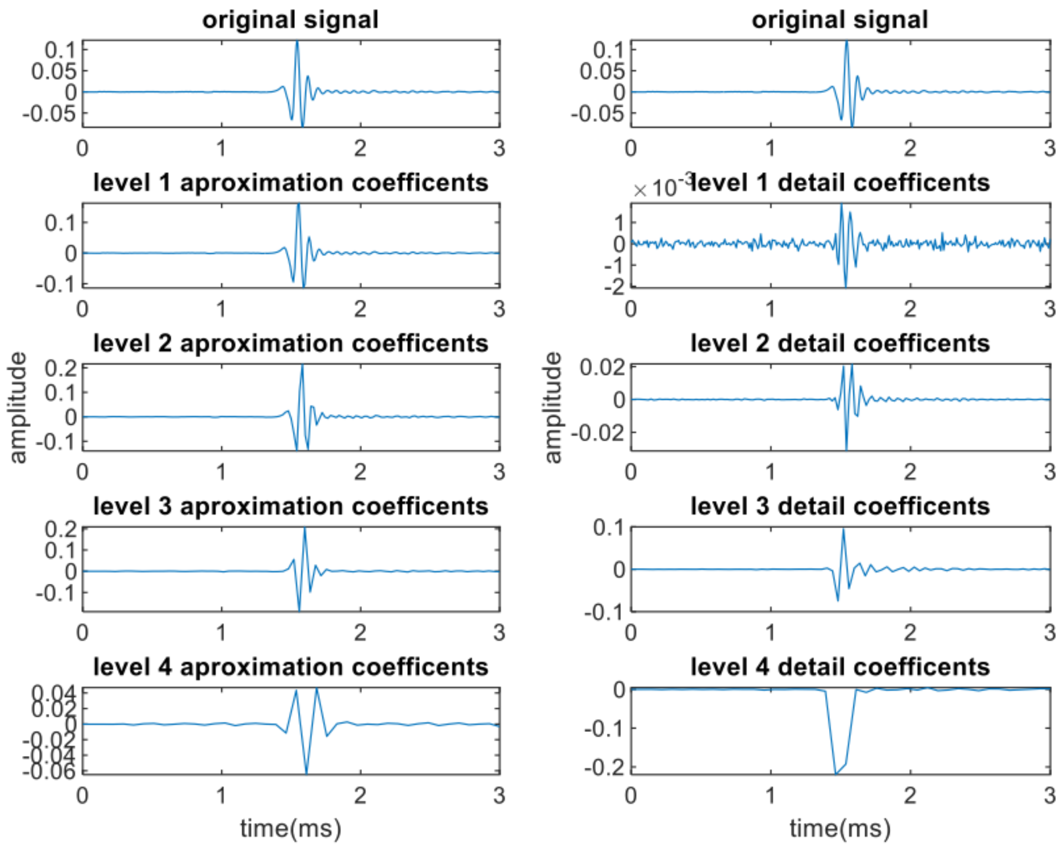

Figure 3: The left column displays the approximation coefficients, while the right column exhibits the detail coefficients. The top plot in each column represents the original signal, as given in Leftwich.[4]

As the decomposition levels increase, the approximation coefficients progressively smooth the signal. This smoothing is evident not only in the approximation coefficients but also in the detail coefficients. The differences between adjacent data points, captured by the detail coefficients, diminish as the decomposition progresses, indicating a smoother signal.

In this paper, our approach involves using the DWT to extract and remove the detail coefficients, resulting in data compression. We will then proceed to reconstruct the data using the remaining approximation coefficients. The main objective is to compare the outcomes of performing the decomposition using various mother wavelets. To determine the optimal wavelet, we will employ a decision tree machine learning methodology and assess its performance against different evaluation metrics.

3. THE CROSS CORRELATION

To find the optimal wavelet for signal compression, we employed the cross correlation technique, which quantifies the similarity between two signals. For a pair of signals, denoted as $f(t)$ and $g(t)$, the discrete cross correlation is mathematically defined as:

$$f \star g = \sum_{i=0}^{N+M-1} f_t \cdot g_{t+\tau} \tag{8}$$

Where N signifies the quantity of data points in signal f, while M denotes the number of data points in signal g.

Essentially, this procedure involves sliding one signal, $g(t)$, along another signal, $f(t)$, and multiplying the two signals together at each time point τ. Larger resultant values indicate greater similarity between the signals. When the two signals are identical, the process is termed autocorrelation, leading to a symmetric outcome around the origin and a maximum value at the origin.

A. USING THE CROSS CORRELATION TO FIND THE BEST WAVELET

In order to use the cross correlation to find the best wavelet for compression of a signal we decompose the signal with a wavelet transform which produces two sets of coefficients. The approximation coefficients and the detail coefficients. Then the detail coefficients are set equal to zero, and the signal is reconstructed by performing the inverse discrete wavelet transform (iDWT) with only the approximation coefficients. This reconstructed signal is then cross correlated with the original signal and the central value is used to determine the similarity between the two signals. A higher central value corresponds to a more similar signal.

For each signal we did this with all available discrete wavelets in the Pywavelet[1] python module. In total there are 106 discrete wavelets available in the Pywavelet module.

As an example we will use the cross correlation of one of the whale clicks. The cross correlation method selected the Discrete Meyer wavelet as the best wavelet for compression and the Biorthogonal 1.1 wavelet as the worst wavelet. The two wavelets are shown in figure 4.

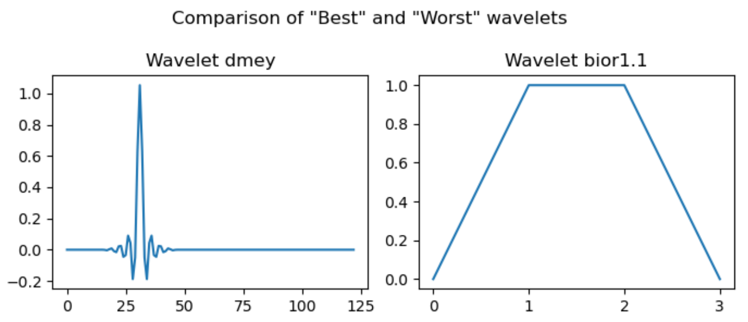

Figure 4: Left: The Discrete Meyer wavelet was chosen as the best wavelet for compression for this signal Right: The Biorthogonal 1.1 wavelet was chosen as the worst wavelet for compression for this signal

In the figure below we show an example of how well this method works for finding the best wavelet for audio compression by comparing the original signal, the signal after reconstruction with the Discrete Meyer wavelet, and the signal after reconstruction with the Biorthogonal 1.1 wavelet.

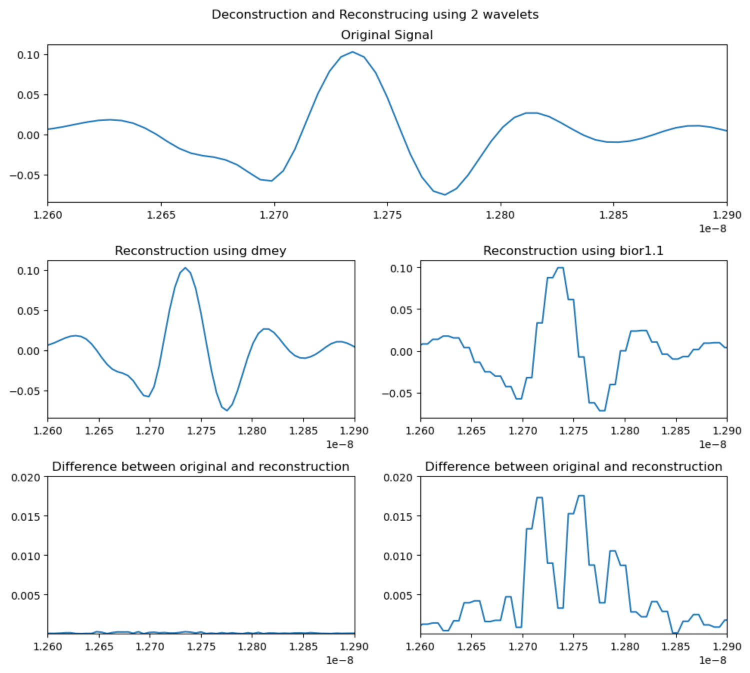

Figure 5: Comparison of cross correlation for determining the best wavelet:
Top: Original signal zoomed in to show details
Left: Reconstruction using the best wavelet as determined by the cross correlation. The central value of this correlation was 0.0933
Bottom: Difference between original signal and reconstruction
Right: Reconstruction using the worst wavelet as determined by the cross correlation. The central value of this correlation was 0.0900
Bottom: Difference between original and reconstruction

The top plot is the original data. In this case it is a sperm whale click taken from hydrophone readings in the northern Gulf of Mexico. As can be seen the wavelet that was determined to be the best by the cross correlation is very similar to the original wavelet. The wavelet that was determined to be the worst looks very different. It did not smooth the data and in fact did the opposite.

Next we wanted to consider the frequency content of the data as the frequency content shows a different aspect of the data. To do this we used the Fourier Transform which is defined as follows:

$$F(s) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi ts}dt \tag{9}$$

or for the discrete version

$$F(s) = \sum_{t=0}^{N-1} f(t)e^{-i2\pi ts/N} \tag{10}$$

Here t represents a discrete time step, s is the frequency, and N is the number of data points in the signal. The cross correlation is then compared in the frequency domain in the below figure by utilizing the Fast Fourier Transform algorithm.[5]

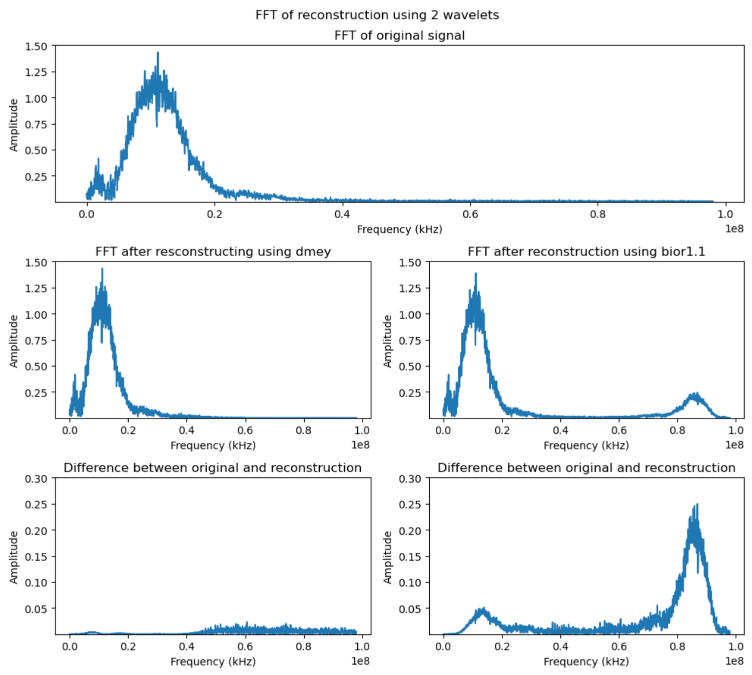

Figure 6: Comparison of signal in Frequency Domain
Top: FFT of original data
Left: FFT of data reconstructed from the best wavelet as determined by the cross correlation
Bottom: Difference between original and reconstructed FFT
Right: FFT of data reconstructed from the worst wavelet as determined by the cross correlation
Bottom: Difference between original and reconstructed FFT

On left side, the original data and reconstructed data look very similar, and the bottom plot shows very little difference in the two signals. The wavelet that produced the lowest cross correlation value, on the other hand actually added frequency content to the signal. This wavelet added higher frequencies that were not in the original signal. The difference plot shows how much frequency content was added.

This method of finding the best wavelet for compression works well. However it does have some drawbacks. Most prominently it is very computationally expensive. For one signal 106 wavelet transforms and inverse wavelet transforms must be calculated, then each of those inverse transforms needs to be cross correlated with the original signal and each central value stored until the entire process has completed. This takes up both memory and computing power and is not a very efficient method for audio compression.

One surprising outcome of using the cross correlation to select the optimal wavelet for signal compression was that although we used all 106 wavelets present in the Pywavelets library on all 302 samples, the cross correlation only selected 12 different wavelets as optimal for compression despite the variety of sounds used.

4. MACHINE LEARNING

The cross correlation was determined to be a useful metric to determine the best wavelet so the next step was to use machine learning on the data we analyzed in order to train a model to find the best wavelet for audio compression. There are many different machine learning algorithms and each one has its own pros and cons. The goal was to find the right method for our purposes. From the cross correlation we had a data set and the best wavelet choice for compression. Each wavelet could be considered a category and so we needed a way to input the data and have the computer choose the best wavelet based on some selected parameters of the original data.

Python has several great machine learning libraries to choose from; SciKit-Learn, Pytorch, and TensorFlow to name just a few. After glancing over documentation for each of these libraries we decided to go with SciKit-Learn for its clear documentation and ease of use. Looking at the different methods available to us we decided to go with a Classification Decision Tree for several reasons: it is easy to understand, easy to implement, and has good visualization.

A. CLASSIFICATION DECISION TREE

A Classification Decision Tree is conceptually easy to understand. It is a machine learning algorithm aimed at classifying input data into distinct categories or classes. This algorithm creates a hierarchical structure as a binary tree, where decisions are based on input features and their correlations to various classes. The process of forming this tree involves choosing input features that optimally differentiate between the classes. Subsequently, these chosen features guide the partitioning of the data into subsets, each corresponding to a branch that emanates from the root node of the tree.

This decision-making procedure is executed recursively for each feature, progressively shaping the tree's layout. An example of this tree structure is shown in the sample tree depicted in Figure 7. The objective of this recursive process is to design a tree that segregates the input data, revealing discernible patterns that lead to accurate classification.

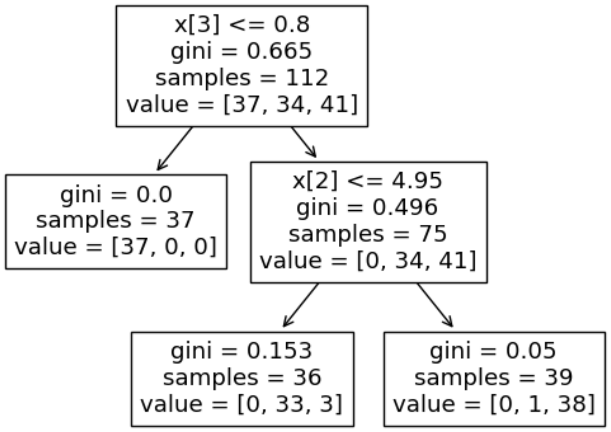

Figure 7: Example Decision Tree. Source: SciKit-Learn[6]
In a decision tree each box that has arrows pointing away is called a node and each box at the bottom, with no arrows, is called a leaf. The example above has two nodes and three leaves. The leaves correspond to the categories or classes that were input into the model
gini is a measure of impurity and ranges between 0 and 1. A gini of 0 means that all of the datasets belong to a single class where a gini of 1 means they all belong to different classes.

When new data is introduced to the algorithm it analyzes the input parameters and compares those to the result. In the example above the tree was made from 112 samples. The value label tells us how many of those 112 samples fit one of the results. For example, the first box says 37 samples fit the first result, 34 fit another, and 41 fit a third result. When new data is input the first thing the computer does is check X[3], the third input parameter. If that value is less than or equal to 0.8 then it sends the data down the left path; in this case it stops there. If on the other hand, X[3] is greater than 0.8, it moves down the right side of the tree and next checks X[2], the second input parameter. Again if it is less than or equal to the stated value 4.95, it goes left, otherwise it goes right. This process continues until it reaches a point where there are no more choices to make.

B. CREATING THE MODEL

In order to create a model to pick the best wavelet for audio compression using the decision tree we chose specific features of the data to feed into the algorithm as input parameters along with the wavelet that was selected by the cross correlation method. This means that the decision tree algorithm could only use 12 of the 106 discrete wavelets available in the Pywavelets library. The input parameters we chose were frequency bins in each signal. For each signal we took the Fourier Transform of the signal and sorted the frequencies into 100 bins. These 100 frequency bins were then used as the input parameters for the decision tree along with the wavelet. The rationale for adopting binned frequency representation in the machine learning model stems from the ability of frequencies to effectively capture and represent signal characteristics. Binned frequency involves categorizing frequency components into discrete bins, facilitating the model's capacity to discern essential frequency ranges that contribute significantly to signal variation.

Once the model has been trained with 80% of the data, then it can be tested with the remaining 20%. The results can then be compared to the cross correlation method described above. We chose to use the standard 80/20 split despite the dataset being relatively small, which could lead to overfitting. This decision was influenced by the fact that the cross-correlation method for determining the best wavelet consistently favored the discrete Meyer wavelet. Consequently, we opted to train the model using a larger portion of the data. This approach aims to expose the model to a broader range of potential outcomes. Simultaneously, we ensured there was sufficient data remaining to thoroughly test the model's performance after its construction.

5. DATA

This project utilized various types of acoustical data to train the machine learning algorithm to choose the best wavelet. All the data consisted of mono .wav files. We incorporated Sperm Whale clicks collected by hydrophone buoys in the Gulf of Mexico (GoM) in the past by the Littoral Acoustic Demonstration Center - Gulf Ecological Monitoring and Modeling (LADC-GEMM) Research Project, as well as audio files of individuals speaking (with their consent) that were recorded more recently. Figure 5 shows a representative sperm whale click collected from the GoM by LADC-GEMM.

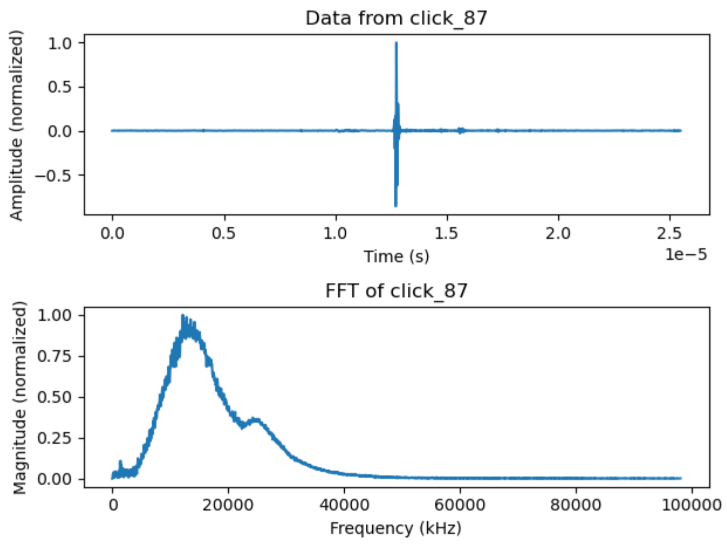

Figure 8: Top: time series click of a sperm whale; Lower: Magnitude of the FFT. Data courtesy of LADC-GEMM

To further diversify the dataset, we included files sourced from freesound.org. Freesound serves as a repository for audio samples uploaded by its users under the creative commons license. It offers a wide array of sound types. For this project, we randomly selected .wav files from the website, encompassing ambient noise, conversations, and a variety of instrumental sounds. Figure 6 is an audio file from www.freesound.org.

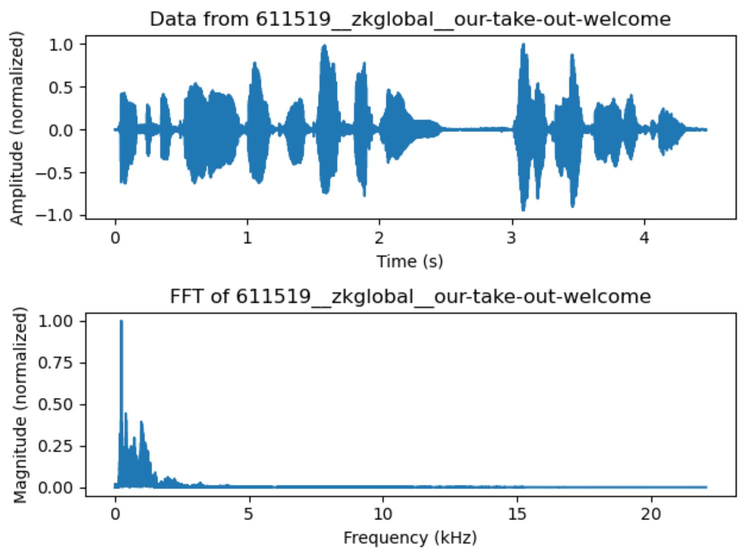

Figure 9: Top: Time series of an audio file from www.freesound.org; Lower: Magnitude of the FFT

A total of 302 signals were employed, comprising 181 whale clicks from the Gulf of Mexico, 21 self-generated audio recordings, and 100 samples sourced from www.freesound.org. The data was used to train the machine learning algorithm to choose the best wavelet for audio compression. The algorithm used was a decision tree, described above.

6. ANALYSIS

Once we had the data and the chosen machine learning algorithm we implemented the algorithm to generate the model and test it. For this purpose we used a random sample of 80% of the data to train the model with the remaining 20% of the data to test the model. We used the cross correlation method to find the best wavelet for all 302 audio files. Then, with 80% (241) of those files we input the 100 frequency bins and the wavelet determined by the cross correlation method to train the model.

The decision tree produced by the model is displayed below. It is shown up to the 4th of 8 levels.

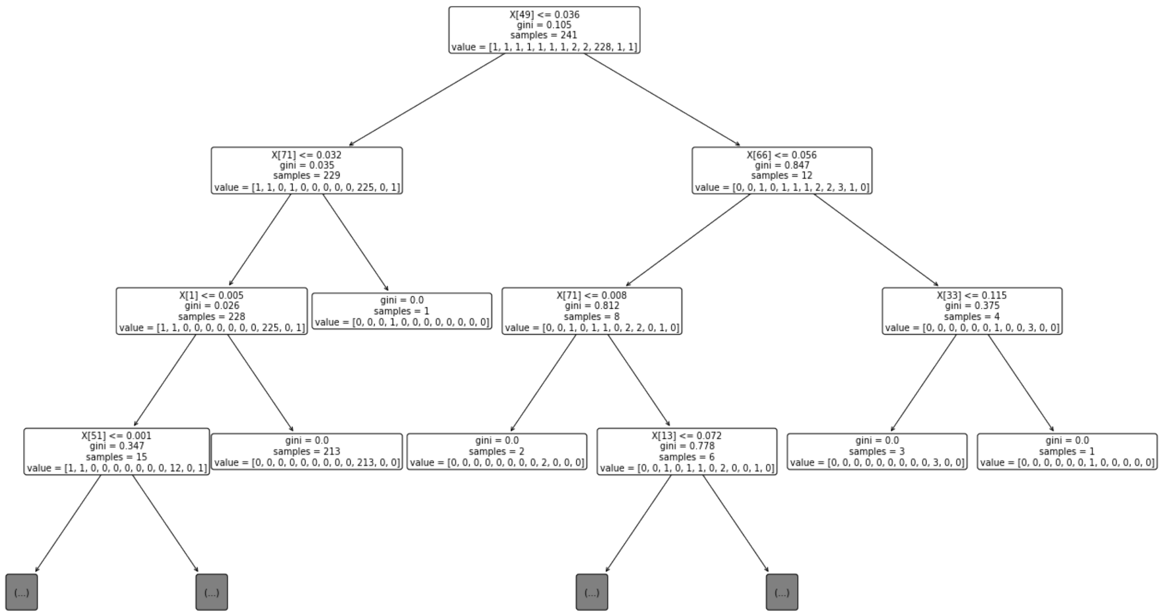
X[49] <= 0.036
gini = 0.105
samples = 241
value = [1, 1, 1, 1, 1, 1, 1, 2, 2, 228, 1, 1]
X[71] <= 0.032
gini = 0.035
samples = 229
value = [1, 1, 0, 1, 0, 0, 0, 0, 0, 225, 0, 1]
X[66] <= 0.056
gini = 0.847
samples = 12
value = [0, 0, 1, 0, 1, 1, 1, 2, 2, 3, 1, 0]
X[1] <= 0.005
gini = 0.026
samples = 228
value = [1, 1, 0, 0, 0, 0, 0, 0, 0, 225, 0, 1]
gini = 0.0
samples = 1
value = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
X[71] <= 0.008
gini = 0.812
samples = 8
value = [0, 0, 1, 0, 1, 1, 0, 2, 2, 0, 1, 0]
X[33] <= 0.115
gini = 0.375
samples = 4
value = [0, 0, 0, 0, 0, 0, 1, 0, 0, 3, 0, 0]
X[51] <= 0.001
gini = 0.347
samples = 15
value = [1, 1, 0, 0, 0, 0, 0, 0, 0, 12, 0, 1]
gini = 0.0
samples = 213
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 213, 0, 0]
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0]
X[13] <= 0.072
gini = 0.778
samples = 6
value = [0, 0, 1, 0, 1, 1, 0, 2, 0, 0, 1, 0]
gini = 0.0
samples = 3
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
(...)
(...)
(...)
(...)

Figure 10: Decision tree generated from the model
This figure shows the first 4 layers of the decision tree generated by our model. When new data is introduced to the model the first thing it does is check X[49] which corresponds to the 49th out of 100 frequency bins. If the value is greater than 0.036 the data is then sent down the right path where it checks the 66th frequency bin. Otherwise it is sent down the left path. It does this until it ends up at a leaf which represents one of the classes that is input into the model

As can be seen, the decision tree was created by 241 samples and there were 12 different possible wavelets. However, most of the samples (213), produced of one wavelet, the discrete Meyer wavelet which is shown in Figure 4.

The other wavelets that were selected by the cross correlation method were the biorthogonal 3.1 and several high numbered Daubechies wavelets (33, 34, and 38). Most of the wavelets other than the discrete Meyer had only one sample that correlated with that wavelet.

With the trained model we then tested the model with the remaining 20% (61) audio files and compared the results of the model with the results as determined by the cross correlation method. The model agreed with the cross correlation method for 91.5% of the samples tested.

7. CONCLUSIONS AND FUTURE WORK

The model we created does a reasonably good job of finding the best wavelet for audio compression. It agreed with the cross correlation method for approximately 91.5% of the tested 61 files but required only fraction of the computational power and time.

There are further issues that should be addressed. Out of 106 discrete wavelets only 12 unique wavelets were selected by the cross correlation method. Additionally out of the 302 audio samples the the discrete Meyer wavelet was selected as the optimal wavelet for 213 samples. This could mean that in general the best wavelet for compression may just be the discrete Meyer most of the time. It also means that if a different wavelet is better for compression then the decision tree will have a harder time finding that wavelet because there are simply not enough samples of that wavelet for the machine learning algorithm to choose. This could be addressed by using a larger number of samples and by finding samples that correspond to each of the wavelets. However, since wavelets are not specific to audio signals there may not be audio signals that correspond to each wavelet.

The next steps in this research are to fine tune the model (possibly with more samples). We also plan to try methods other than the cross correlation for selecting the best wavelet prior to implementing the machine learning model, or use a different machine learning approach, such as a convolutional neural network.

Then we will use what we have learned here to apply machine learning to different aspects of signal processing such as feature enhancement or denoising data.

REFERENCES

[1] Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary. Pywavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237, 2019.

[2] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge, 2009.

[3] J.O. Smith. *Spectral Audio Signal Processing*. W3K, 2011.

[4] K.M. Leftwich. *Denoising and Deconvolving Sperm Whale Data in the Northern Gulf of Mexico using Fourier and Wavelet Techniques*. PhD thesis, University of New Orleans, December 2022. https://scholarworks.uno.edu/td/3051.

[5] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[6] scikit-learn: Machine learning in python. https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html#sphx-glr-auto-examples-tree-plot-unveil-tree-structure-py. Accessed on 5/24/2023.