CSC 869
Term Project
Classifying Heartbeat Sounds

By:
Ali Alavi

Table of Contents

Problem Statement

Cardiovascular diseases are one of the leading causes of death worldwide according to WHO. Therefore, being able to detect abnormalities in heartbeats is an important task. The motivation behind this project is to be able to automatically classify and distinguish normal heartbeats from abnormal ones.

The proposed solution should be a Machine Learning Model that can classify real heart audio (also known as “beat classification”) into one of multiple categories: Normal, Extrahls, Extrastole, Murmur, Artifact. Heartbeats consist of 2 consecutive noises called ‘lub’ and ‘dub’ which are produced due to the heart movements. These noises are of varying frequencies and occur in close proximity in normal situations. In my project, I aim to use different techniques to classify these heartbeats into the aforementioned classes.

The Data

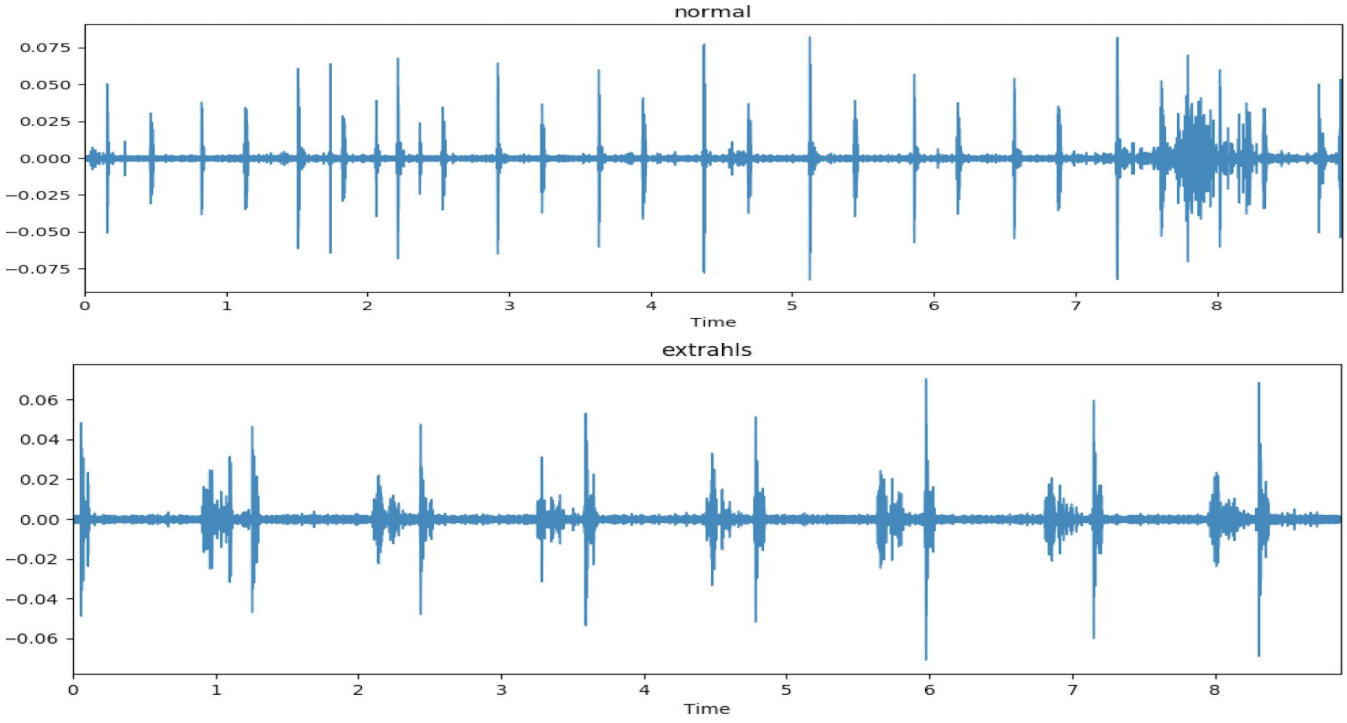
The data was available on Kaggle in the following format

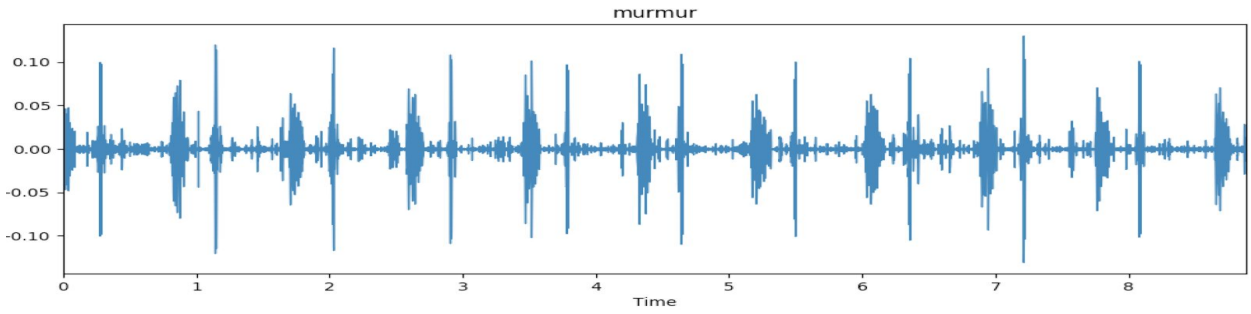
- set_a.csv - Labels and metadata for heart beats collected from the general public via an iPhone app
- set_b.csv - Labels and metadata for heart beats collected from a clinical trial in hospitals using a digital stethoscope
- audio files - Varying lengths, between 1 second and 30 seconds. (some have been clipped to reduce excessive noise and provide the salient fragment of the sound). They are represented in the .wav format.

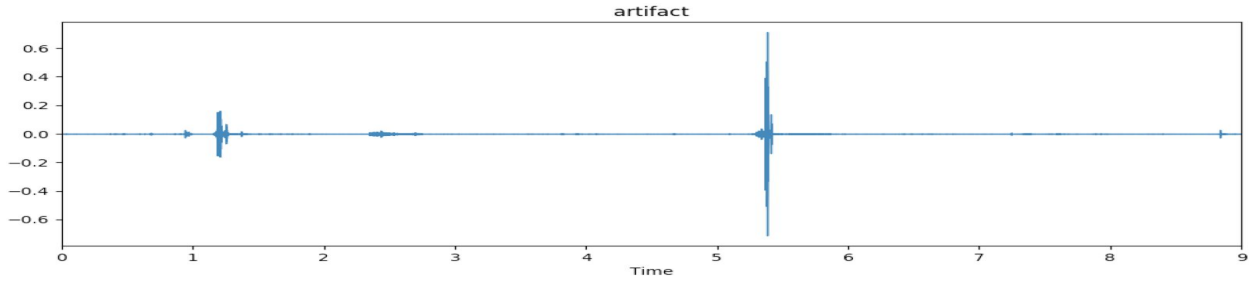
The data can be downloaded from https://sfsu.box.com/s/zg5sroodcbjgzux7x8tedxxncb6cqsuv

Visualizing The Data

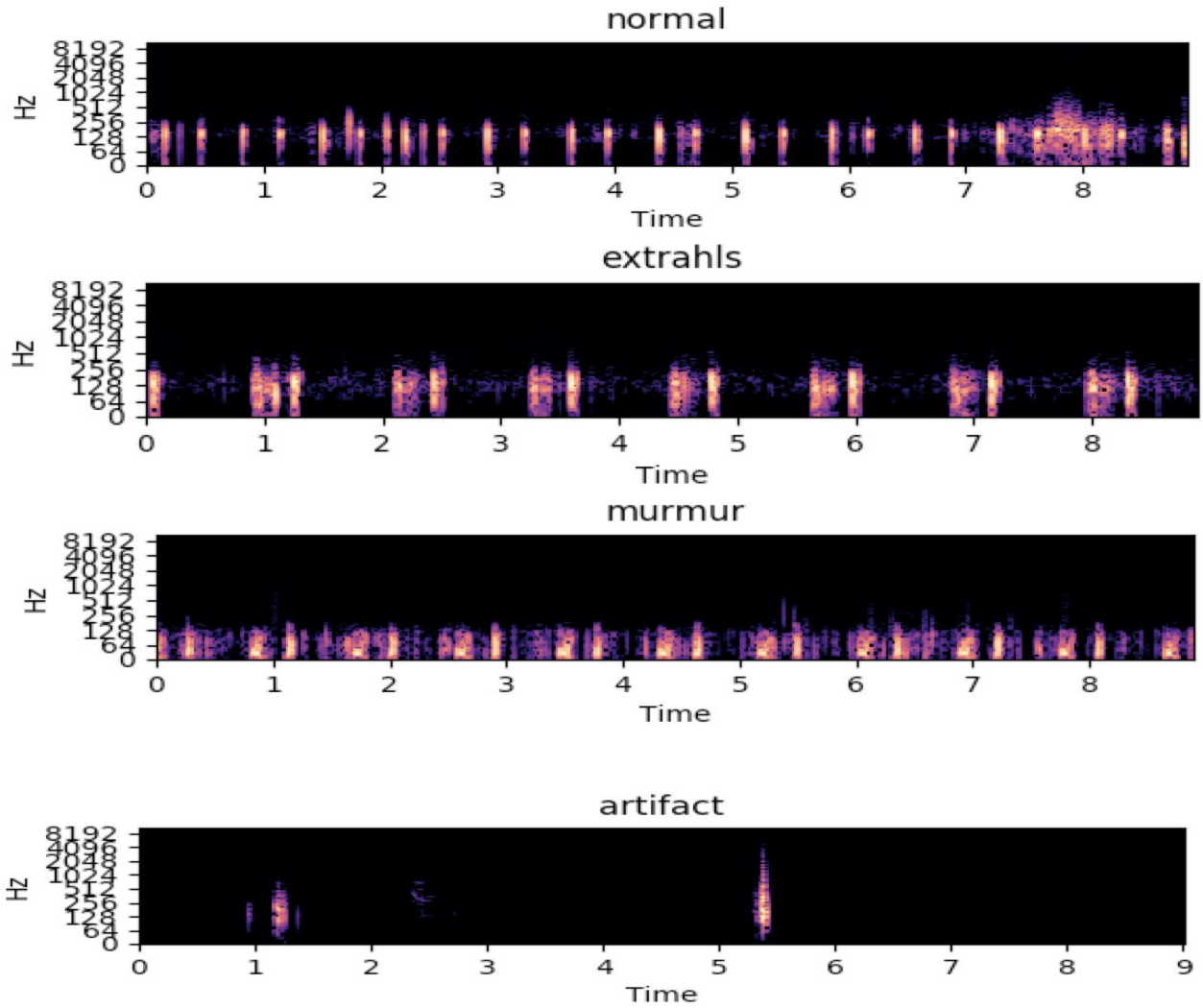
Waveplots







Spectrograms



This basic visualization was very helpful in understanding the data I was dealing with. For example the waveplot shows the amplitude of the audio signals whereas the spectrogram displays a spectrum of the frequencies present in the different classes.

Data Problems, Feature Extraction, and Preprocessing

The only major issue with the dataset was that there were no truth labels for test data in both datasets, A and B. Due to lack of domain knowledge, I was unable to label them on my own but upon researching, I discovered that a Doctor had provided labels for all the files in set A. He had even relabeled the whole dataset which I represent in the column called integer_labels in the updated version of the CSV file. He even removed one class and decided to have only 3 classes for dataset A: 0=artifact, 1=normal/extrahls, and 2=murmur.

I had no such luck for for dataset B, so I removed the unlabled files and their corresponding entries in the CSV file and split the training data into training and test data randomly. This resulted in 404 files for training and 57 files for testing. I used some Python code to update the CSV file and add the integer_label column using 0=extrastole, 1=normal, and 2=murmur.

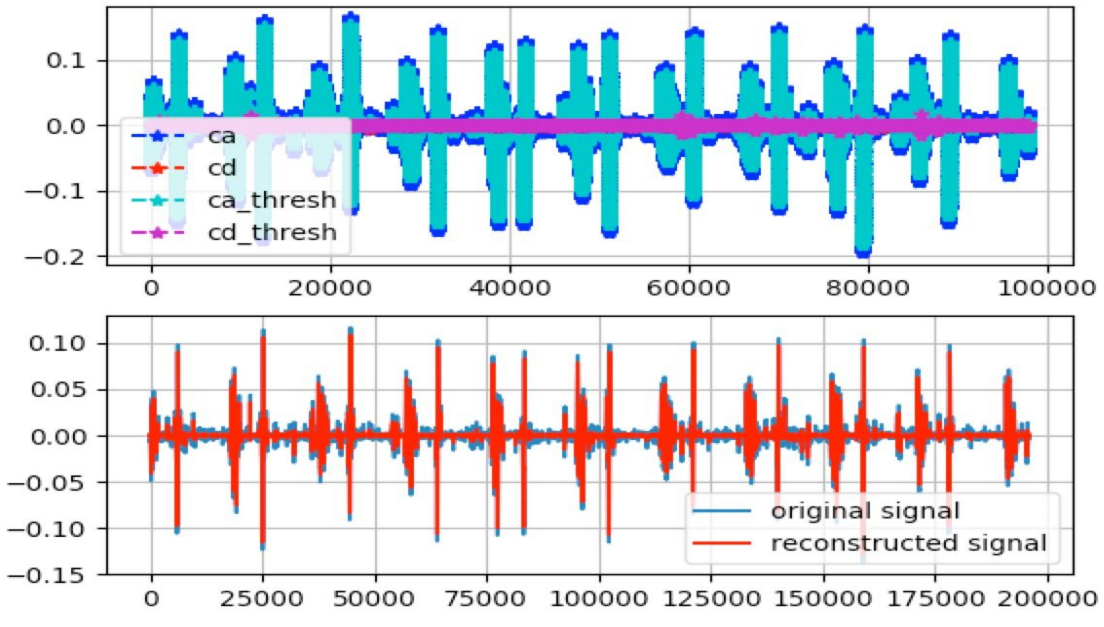
For feature extraction, I used Librosa to extract the following features from the audio data:

stft = Short-time Fourier transform
mfccs = Mel-frequency cepstral coefficients
chroma = Compute a chromagram from a waveform or power spectrogram
mel = mel-scaled spectrogram.
contrast = Spectral Contrast
tonnetz = tonal centroid features

Wavelet Transformation

Wavelet transformation is a method used to break down a signal into a series of coefficients. The first coefficients represent the lowest frequencies, whereas the last coefficients represent the highest frequencies. The signal can be smoothed by removing the higher frequency coefficients and then reconstructing the signal with the truncated coefficients, unlike using a moving average where we smooth over all of the peaks that are of interest.

I have used the PyWavelets module to remove noise from signals. An example of reconstructing the signals for a murmur audio from the dataset can be seen below



Classification Approaches

For classifying the data, I applied 3 different approaches: Neural Net using Tensforlow, Neural Net using Keras, and Decision Tree using sklearn libraries.

For the Tensorflow and Decision Tree approach, I used the above mentioned features librosa whereas for Keras, I used the audio data points represented as a numpy array given by loading it via scipy's wavfile library. To make sure that the number of features are same, I used a repeat length function to repeat the array for each audio to the longest audio file from that data set.

For Tensorflow, the 2 layers used were Tan and Sigmoid, Cost functions ReduceMean over ReduceSum and ReduceMean over Softmax Cross Entropy, Optimizers Gradient Descent and Adam. Set A consisted of 124 files for training and 52 files for testing. Set B consisted of 404 files for training and 57 files for testing.

For Keras, I used 15 layers which consisted of Convolutional, MaxPool, BatchNormalization, Dropout, GlobalAvgPool and Dense. The data was split randomly using Sklearn to get 70% data for training and 30% for testing for both sets.

For Decision Tree, I used Entropy and Gini Index as splitting functions. Set A consisted of 124 files for training and 52 files for testing. Set B consisted of 404 files for training and 57 files for testing.

Results

The following accuracies were achieved by running the different approaches on the 2 datasets with and without wavelet transformation

1- Tensorflow

With 500 epochs

		Set A		Set B	
Cost Function	Optimizer	Normal	Wavelet Transform	Normal	Wavelet Transform
Reduce Mean over Reduce Sum	Gradient Descent	82.7	65.4	49.1	43.9
Reduce Mean over Softmax Cross Entropy	Adam	80.8	80.8	56.1	54.4

2- Keras

With 30 epochs, and 100 steps per epoch

		Set A		Set B	
Cost Function	Optimizer	Normal	Wavelet Transform	Normal	Wavelet Transform
Categorical Cross Entropy	Adam	84.62	80.25	70.37	66.75

3- Decision Tree

	Set A		Set B	
Split function	Normal	Wavelet Transform	Normal	Wavelet Transform
Entropy	75	69.23	36.84	49.12
Gini Index	71.15	69.23	49.12	49.12

As we can see, Set A had much better results than set B. This could be because of the relabeling done by the domain expert which was helpful in building the classifiers whereas set B could have bad labels. Keras's multi layered Neural Net was the best approach at classifying both datasets A and B and the Adam optimizer performed well for both Keras and Tensorflow. Decision Trees performed poorly on both data sets.

In addition, it can be observed that performing a wavelet transformation resulted in a degraded performance for all approaches which could be due to the lack of noise present in the data sets.

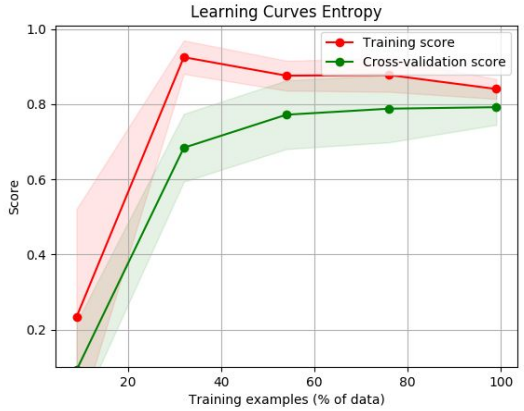
Learning Curves

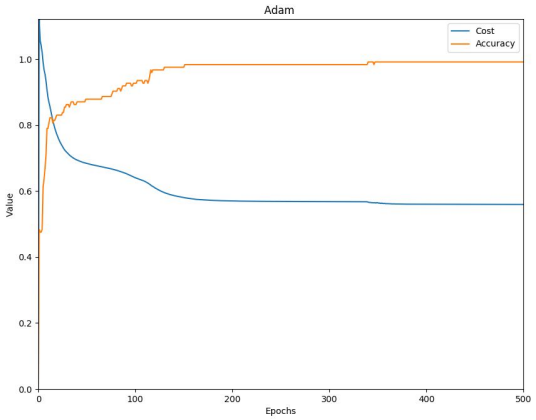
Plotting learning curves is a good way of showing whether there is variance or bias in our classifiers.

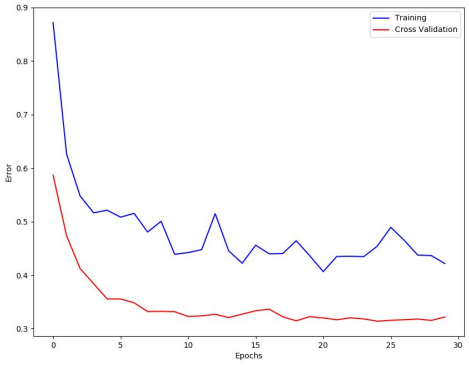
If the gap between the training and cross validation error is negligible as the data size grows, it means that the classifier is suffering from high bias aka underfitting. In this case adding more items to the training set would not be helpful as the error rate stays similar.

If the gap between the training and cross validation error is high as the data size grows, it means that the classifier is suffering from high variance aka overfitting. In this case adding more items to the training set would be helpful as the cross validation error appears to be coming down.

In addition to the ones displayed below for D trees, Tensorflow and Keras, I have provided the learning curves with the submission in a folder titled learning curves with the file names representing the approach, data set used and whether wavelet transformation was used or not for that instance







Pros and Cons

As evident from the results, the biggest con of my approach is the size of dataset with proper labels. Set A was limited in the amount of data available whereas set B was missing labels for test sets and probably had poorly labeled files. The pros are that I was able to achieve really good results on set A using Neural Nets and this is promising for classification in the similar domain.

Conclusion

To conclude, using Neural Nets to classify audio files is a promising approach as we have seen in this project. Set A which consisted of audio collected from the iPhone app was remarkable and in the future, using an approach like mine, we can provide real time feedback to the user about potential abnormalities in their heartbeats. It would also be helpful to find better ways to label heartbeats by getting more insight from a domain expert i.e. cardiologist to correctly label the training and test data.