5-1-2017

COMPARISON OF EMBEDDED AUDIO SIGNAL CLASSIFICATION METHODS USING RULE-BASED ALGORITHMS AND ARTIFICIAL NEURAL NETWORKS

Bradley Wabbersen

COMPARISON OF EMBEDDED AUDIO SIGNAL CLASSIFICATION METHODS USING RULE-BASED ALGORITHMS AND ARTIFICIAL NEURAL NETWORKS

by

BRADLEY WABBERSEN

Presented to the Faculty of the Honors College of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

HONORS BACHELOR OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2017

ACKNOWLEDGMENTS

First, I would like to thank the faculty and staff of the Honors College at the University of Texas at Arlington for all their support during my undergraduate career and for pushing me to get more from my education.

I would also like to thank Dr. George Kondraske, my faculty mentor for this project, for all his guidance and support.

I want to extend special thanks to all the professors who have challenged and inspired me over the past four years. I especially want to thank Dr. Steven Gibbs, Dr. Jason Losh, Dr. James Warren, Dr. Venkat Devarajan, and Dr. Howard Russell. You have truly made an impact.

Finally, thank you to my family and friends who have been a source of strength and encouragement through the past four years.

May 5, 2017

ABSTRACT

COMPARISON OF EMBEDDED AUDIO SIGNAL CLASSIFICATION METHODS USING RULE-BASED ALGORITHMS AND ARTIFICIAL NEURAL NETWORKS

Bradley Wabbersen, B.S. Electrical Engineering

The University of Texas at Arlington, 2017

Faculty Mentor: George Kondraske

Accurate and efficient audio classification algorithms have significant applications in the modern world. Song recognition and speech recognition apps rely on complex audio processing software. Recent research in this field has focused primarily on the application of artificially intelligent systems to solve difficult audio processing problems where strictly rule-based algorithms would be untenable. The problem to be investigated in this project is a much simpler binary classification scenario, but with a much smaller sample set than is typically used for neural network training as well as limited processing resources. It will be determined whether in such a scenario a neural network approach will still outperform a strictly rule-based algorithm. Each implementation is given identical sound samples to test against. Samples include a collection of desired and undesired sounds. Accuracy is

measured as a percentage of how many test samples are correctly categorized in each category. The results of this testing demonstrate that a neural network based classifier outperforms a rule-based classifier in terms of overall accuracy in this scenario.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Problem Context

1.1.1 Problem Statement

This paper focuses on the research and development involved in a binary audio classification system. A signal must be classified as either a desired or undesired signal with a high degree of accuracy. This problem can be approached in two distinct ways. The conventional method would be to develop a set of rules to decide whether a signal can be classified as a desired signal. Rules could be applied to signal characteristics such as spectrogram coefficients or bandwidth. Strictly rule-based algorithms use deterministic mathematical functions and logic statements to solve problems. Alternatively, many modern audio processing algorithms use artificial neural networks (hereafter *neural networks*) to solve the problem. Neural networks fall into the rapidly growing field of artificial intelligence. Neural networks can yield more accurate results compared to strictly rule-based approaches but are not always practical to the situation. Additionally, many engineers are reluctant to use neural networks because they do not follow rules that humans can easily understand.

This Honors College project was conducted within the context of an electrical engineering senior design group project. The senior design project sponsor was Dr. Greg Turner from the Department of Electrical Engineering. The objective of the group project was to create a functioning prototype for a dog door that opens automatically in response

to a given dog bark. The senior design project aimed to produce a prototype as close to a possible retail product as design time allowed.

The audio classification system was required to classify incoming audio signals as either a desired dog bark that would open the door, or as an undesired sound that would be ignored. A collection of audio recordings of Dr. Turner's dog were provided by Dr. Turner. Audio recordings of a specifically trained bark were classified as "good barks." Other untrained (defensive or aggressive) barks from the same dog were classified as "bad barks." Both a rule-based classifier and a neural network classifier were implemented and tested. This report will compare the results of the two methods.

1.1.2 Problem Constraints

A constraint placed on the senior design project was to use the Raspberry Pi model 3B as the main processor for the audio classification system. The Raspberry Pi runs on a Linux operating system and has many of the features of a PC in a small package. Python is perhaps the best programming language for working with the Pi in terms of both cross-platform compatibility and library support. Thus, all the algorithms discussed in this paper are implemented in Python.

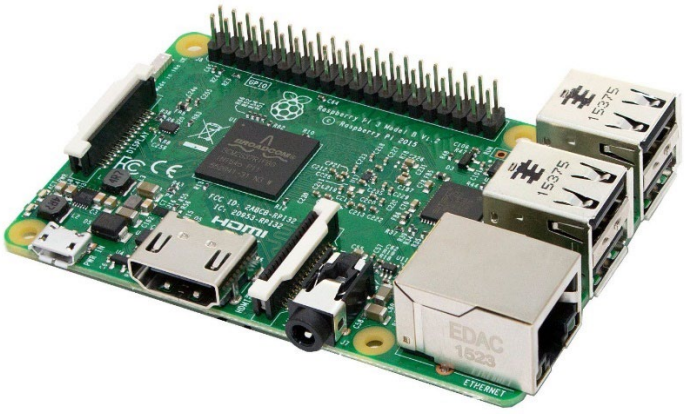


Figure 1.1: The Raspberry Pi 3B [1]

1.2 Background

The sound classification problem is certainly not new. Significant research has already been conducted on methods for categorizing sound samples. Both traditional and artificially intelligent systems have been tested and compared. More recent research tends to favor some kind of artificial intelligence approach. [7][8][9][10][11] This particular problem is similar to previous research problems but has a few unique factors. Neural networks usually prove most effective in cases where the problem to be solved is very complex and sufficient test data is available. The relative simplicity of the binary categorization as well as the limitations of a small sample set may mean that a rule-based system will perform better.

1.2.1 Neural Networks

The idea behind neural networks is loosely based on how the human brain processes data. Neural networks work well in non-deterministic systems such as computer vision and speech recognition. Speech recognition apps like Siri or Alexa are an example of how neural networks can be used in modern applications. Neural networks use a large interconnecting group of processing units called neurons to solve complex problems. Each neuron sums together all of its inputs, applies a bias and a non-linear activation function, then passes the data to the next neuron or neurons.

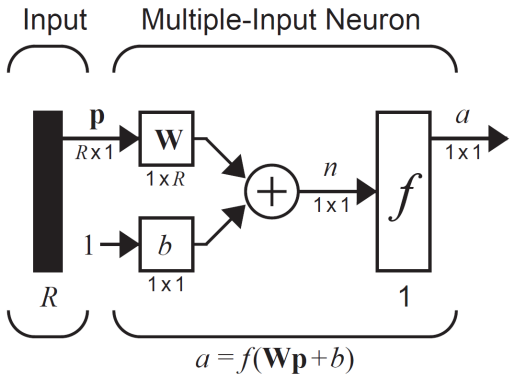


Figure 1.2: A Single Neuron [2]

These neurons are typically grouped into processing layers that form a path between data input and output.

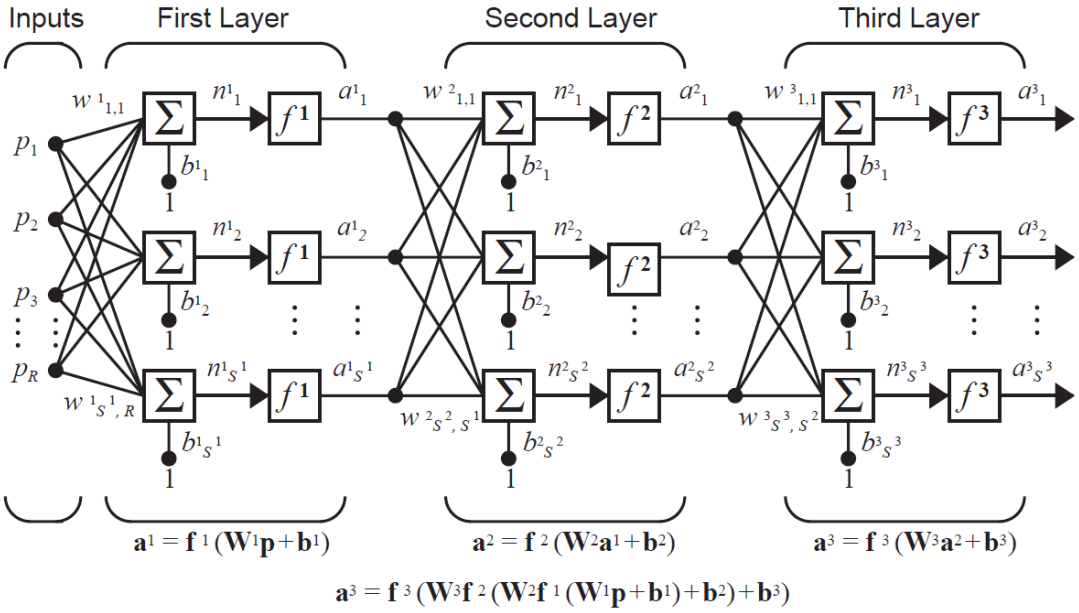


Figure 1.3: A Three Layer Feedforward Network [2]

The above figure illustrates a type of network architecture called a feedforward network. In this architecture, data flows in one direction from input to output. Other architectures exist, such as recurrent networks, where data paths can form loops. Because of the kind of data involved in this particular problem, such networks would not be advantageous. Thus, this paper will focus on feedforward networks.

The specific network architecture can be customized to address a specific problem. The network weights and biases are determined by a training algorithm. Possible training methods can be separated into three categories: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, an input is provided to the network and a desired output is also provided. Unsupervised learning and reinforcement learning are used in cases where no desired outputs are available for training. They are typically used in control systems or clustering applications. In this problem, a set of audio samples and their associated classifications (1 – "Good Barks", 0 – "Bad Barks" and any other audio samples) is available for training. Thus, this paper will focus on supervised learning.

During training, an optimization technique iterates through possible network values until the network can satisfy the training sample set with acceptable accuracy. Even for simple networks, this can take many thousands of iterations. One of the most widely used optimization methods is gradient descent. [4] Gradient descent tries to minimize a loss function (in this case the accuracy of the network to known sample classifications) by moving network values in a direction that reduces the loss function. Typically, samples are separated into two groups: training samples and evaluation samples. After a training session, overall network accuracy can be measured by running the evaluation samples through the network and comparing expected values to actual values produced by the network. This is important because a network can be easily trained to correctly categorize samples it has seen before. The real test of accuracy is in measuring the accuracy for new samples.

1.2.2 Convolutional Neural Networks

Convolutional neural networks are a special kind of feedforward network that implement a convolutional layer. Convolutional networks were inspired by the visual cortex of the brain; the area responsible for image processing. They are most commonly used in image and video applications but will also be useful for this project. One big advantage of CNNs is that they are translation invariant, meaning the input data can be shifted in along any dimension and the CNN will still be able to detect the relevant patterns. Convolutional layers are formed by an array of filters which scan the input space. These filters are trained to recognize distinguishing features in the input space.

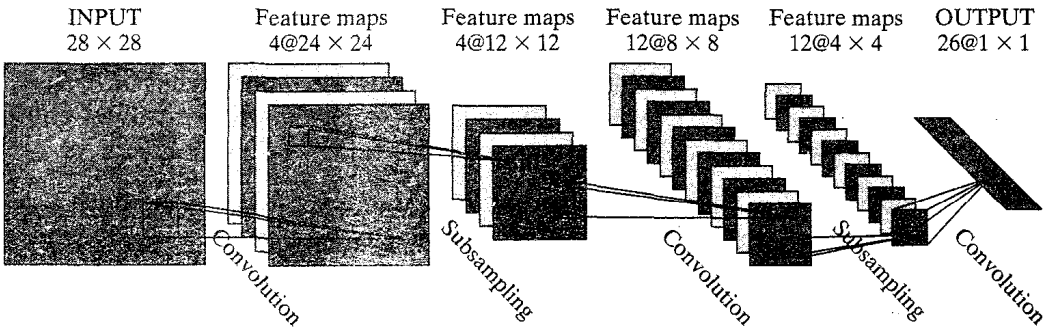


Figure 1.4: Example of a Convolutional Network [3]

CHAPTER 2

METHODOLOGY

2.1 Preparing the Data

2.1.1 Expanding the Sample Set

The original sample set consisted of the dog bark recordings provided by Dr. Turner. This included recordings of both "good barks" and "bad barks." As a general rule, the sample set should be as large as possible to effectively train a neural network. For the purposes of neural network training, it is possible to expand the sample set artificially by copying the original samples and introducing noise. Each of the original samples was copied in this way a total of six times by introducing white noise, pink noise, and blue noise at SNR levels of 10dB and 20dB. In total, this resulted in each category having about 50 samples each.

In addition to the recorded dog barks, another set of recordings was added to the sample set. Several noises were recorded including table banging, door slamming, and human imitations of a dog bark. Additionally, the urban sound dataset compiled by New York University [5] was included. The urban sound dataset included over 8000 samples from ten categories including air conditioners, car horns, children playing, dog barks, drilling, engine running, gun shots, jackhammers, sirens, and street music.

The additional recordings are necessary to the development of the audio classification system. Because the input to the classifier could be any sound, the classifier must be able to correctly classify any sound other than a "good bark" as a negative.

2.1.2 Conforming the Sample Set

Before any kind of audio classification can take place, data must be converted into a common format. All files were converted to wav files sampled at 48000Hz at 32 bit depth. The samples from Dr. Turner have useful audio information for a duration of approximately 500ms. Thus, it was determined that all samples must be clipped to a length of 500ms. Any samples shorter than 500ms were zero padded to 500ms. In order to clip out the desired audio segment, a program was written to detect the first amplitude spike and extract audio 100ms before and 400ms after the spike.

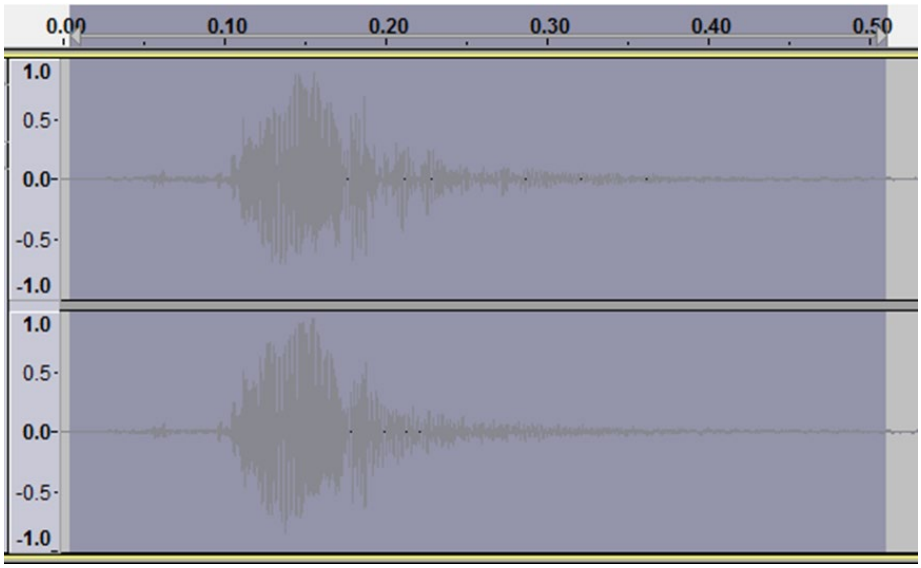


Figure 2.1: Waveform Example for a Dog Bark

2.1.3 Feature Extraction

The raw audio waveform itself is not very useful for audio classification. Useful features must be extracted before classification can take place. Possible features were selected based on features used in similar research. [7][8][9][10] Additionally, features were tested on the dog bark samples from Dr. Turner to see whether the resulting features would be useful. Features were determined to be useful if they showed a low deviation

between samples in the same group and a high difference between samples in different groups.

Nine features were chosen:

- The Mel-scaled spectrogram coefficients (MFSC)
- The first delta of the MFSC
- The second delta of the MFSC
- The x-axis autocorrelation of the MFSC
- The y-axis autocorrelation of the MFSC
- The spectral contrast
- The spectral centroid
- The spectral bandwidth
- The RMS energy

An audio sample can be broken up into its frequency components via a Fourier transform. If the audio sample is first broken up into smaller sub-windows, the frequencies can be shown to change with time. This time-frequency relationship is called a spectrogram. It has been shown that human hearing is non-linear, both in amplitude detection and in frequency discrimination. The log-scaled Mel spectrogram compensates for these non-linearities, emphasizing components most recognizable to human hearing. Most audio classification research uses the Mel spectrogram as it has been shown to increase the accuracy of models.

Once an audio sample has been loaded, it is broken into 20ms Hanning sub-windows with a 5ms overlap. This results in 101 time-slots and 128 frequency bands. All other features are derived from the MFSC. The Librosa [14] library for Python was used

to create a function to perform this feature extraction process on wav files. Each of the classification methods use an identical feature vector from this function as input.

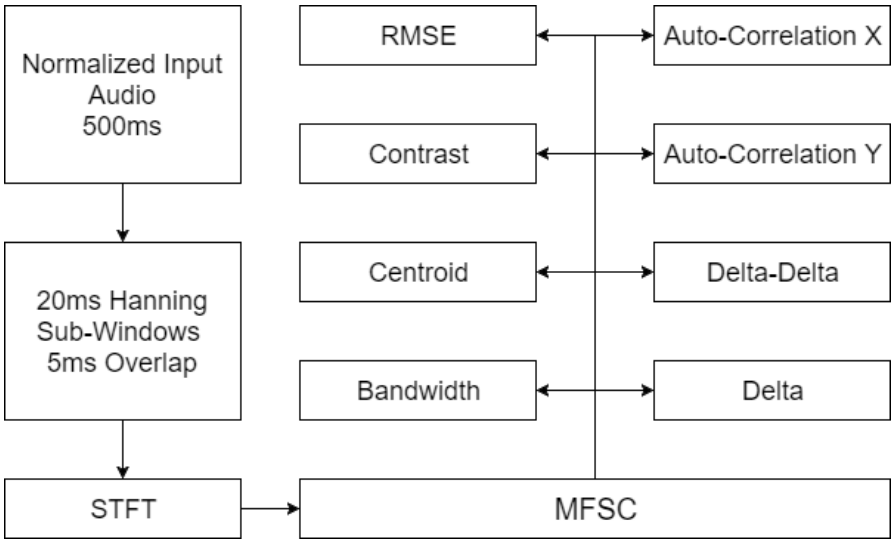


Figure 2.2: Feature Extraction Flow Diagram

The resulting feature vector is a 128 by 101 by 9 matrix. Because the MFSC is 128 by 101, the delta and autocorrelation features will also be 128 by 101. The other features have the same length along the time axis but do not have the same frequency band dimension. These features are zero padded to fill up the 129 by 101 space. All of the features are stacked to produce the 128 by 101 by 9 feature vector. Organizing the data in this way will make the data better suited for input to a convolutional neural network. The data can be treated in the same way a three channel RGB image would be treated; the only difference being that the feature vector has nine channels.

2.2 Classification

2.2.1 Rule-Based Classification Methods

The rule-based approach that was developed relies on comparison. The dog bark samples were run through feature extraction and the average of the features was computed. From that, the relative deviation was also computed. The results are shown in Appendix A.

Using these values, a comparison test was created. The equation for this test is shown below:

$$deviation = \sum_{z=0}^{f} \sum_{x=0}^{n} \sum_{y=0}^{m} \left|A_{x,y,z} - X_{x,y,z}\right| \cdot \left[1 - \left|D_{x,y,z}\right|\right]$$

This equation represents how much the new sample differs from the "good bark" pattern. *X* is a feature from the new sample. *A* is the average for that feature in the "good bark" sample set. *D* is the relative deviation for that feature. *A*, *X*, and *D* are all matrices normalized to 1 with dimensions *n* by *m* by *f* feature layers.

A rule-based classifier, then, would compare the output deviation of the equation above with a scalar threshold hardcoded into the design based on testing. By running samples through this equation, a pattern can be established for "good barks" and all other sounds. Samples that are classified as "good barks" will tend to be below a certain value deviation while all other samples will tend to be above it. Through this analytical process, a scalar threshold can be chosen.

The theory behind this comparison algorithm is that "good bark" features will tend to be similar to the average for that category, and thus *A*-*X* will tend to be small. Furthermore, any elements of the feature layers which display a high deviation from the average will be minimized by multiplying *1*-*D*. This ensures that only feature elements

which are characteristic of the “good barks” sample set will be used to determine whether a new sample falls within that category.

Using this basic equation, three comparison tests were created. The simplest case merely compares the frequency components. To do this, the spectrogram feature is summed along the time axis before being fed into the comparison test. This is equivalent to simply taking the Fourier transform of the signal.

In the second case, the entire spectrogram feature is used for comparison. Finally, in the third case, all features are used for comparison.

2.2.2 Neural Network Classification

Google Tensorflow [13] was used to implement the neural network classification system. Tensorflow can be implemented in Python, making it easy to interface with the rest of the system.

Because each feature is a two-dimensional matrix, each feature can be treated as a greyscale image. The entire feature vector can be treated like a multichannel image. Previous research shows that convolutional neural networks outperform conventional feedforward networks for audio classification where features are input in this way. This makes sense as convolutional neural networks are usually designed to work with multicolor images.

A second order convolutional network was designed, taking inspiration from one of Google’s example neural networks for image classification. This design has two convolutional layers, two pooling layers, a fully connected layer, a dropout layer, and two output neurons that produce the probability that a given sample is in each category. The network returns the category with the highest probability.

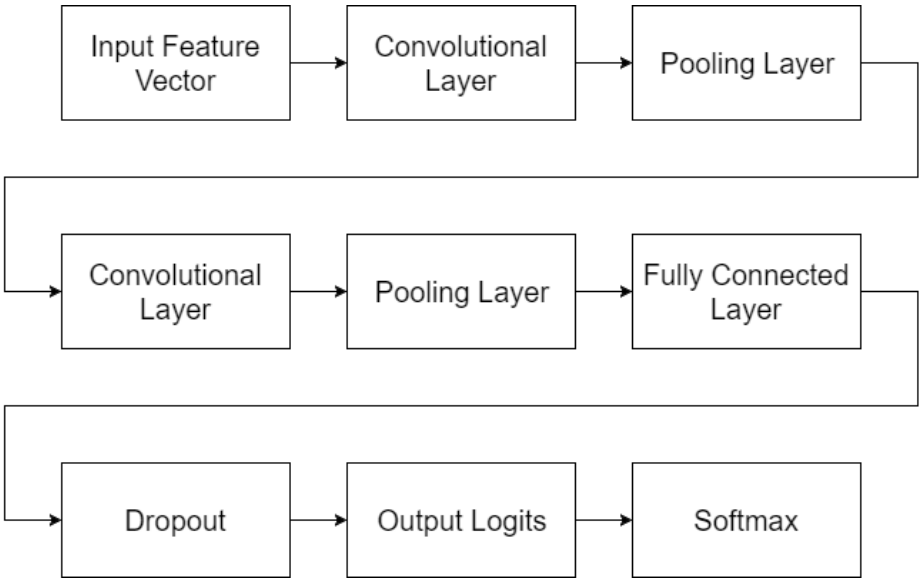


Figure 2.3: Convolutional Neural Network Flow Diagram

The network is designed to be trained via stochastic gradient descent. Network variables are saved to a hard drive allowing the trained network to be ported to any device without the need to retrain the network. This also allows the network to run additional training sessions with new samples after the initial training without losing progress.

For training, samples were chosen randomly from each of the sample subsets in such a way as to guarantee that each sound was represented with a uniform distribution. Samples were then split into two groups: 75% were used for training, again maintaining uniform distribution, leaving the other 25% for evaluation. During each training iteration, all of the training samples are run through the network. The output probabilities are compared to the sample classifications. The loss function results from the difference between the generated probabilities and the sample classifications. Specifically, loss is computed as the cross-entropy between the probability logits and the sample classifications. The network then attempts to improve on the loss function by changing the

network variables via gradient descent. The network was able to converge to an acceptable loss function within about 20,000 training iterations. This took about 40 hours to complete.

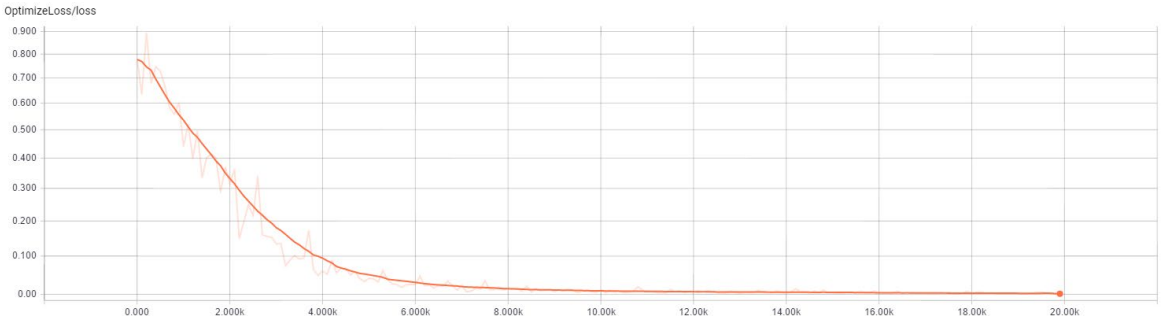


Figure 2.4: Loss Function vs. Training Iterations

CHAPTER 3

RESULTS

Each classification method was tested against identical randomly selected samples from an even representation from each of the sample subsets. In total, 300 samples were chosen; 20 samples per subset. The included subsets were good barks, bad barks, table banging, door slamming, human imitations of dog barks, air conditioners, car horns, children playing, dog barks, drilling, engine running, gun shots, jackhammers, sirens, and street music. The number of correctly classified samples was summed and divided by the total number of samples to get the percent accuracy. The table below shows the accuracy of each classification method for each category of sound processed. “Good Barks” and “Bad Barks” refers to the recordings provided by Dr. Turner and the “Other Sounds” category refers to all other subsets. Thus “Good Barks” and “Bad Barks” each represent 20 samples tests while “Other Sounds” represents 260 sounds tested. This test was repeated several times and recorded in Appendix B. The table below represents the average results.

Table 3.1: Classification Method Accuracy Comparison

	“Good Barks” correctly classified as 1	“Bad Barks” correctly classified as 0	Other Sounds correctly classified as 0	Total Accuracy
Frequency Comparison	100%	100%	95%	96%
Spectrogram Comparison	100%	100%	98%	98%
Full Feature Comparison	100%	60%	99%	96%
Convolutional Neural Network	100%	100%	99%	99%

Because the comparison tests rely on an arbitrary threshold, the threshold must be manually adjusted to get the best accuracy. This is not unlike the way neural networks operate, except that the optimization method is human intervention and only a single variable is used. For this to work, all of the good barks must be below the threshold and all of the other sounds must be above it. A problem occurs when some of the bad samples actually look more similar to the good bark pattern than some of the good bark sounds. The first two comparison tests actually produce surprisingly good accuracy, with the second being more accurate than the first. Generally speaking, more information tends to lead to better accuracy. In the third case, the best accuracy that was achievable was much worse than the other two comparison tests. Upon closer inspection, the additional features did not show sufficient difference between categories once the summation was applied. 40% of the bad barks had feature summations that were lower than the same feature summations on the good barks. No matter where the threshold was placed, one of the categories would have suffered from poor accuracy.

The neural network outperformed all of the comparison methods. This makes sense. Generally speaking, more information leads to higher accuracy when used properly. Exactly how much non-redundant information the extra features provide is a question beyond the scope of this paper. This paper assumes that at least some of the feature information from each feature layer is unique. The third comparison test did not analyze the additional information with a complex enough algorithm for that additional information to be useful. Unlike the spectrogram feature, the other feature layers do not display the same simple patterns between categories. Thus, a simple threshold is insufficient for analysis. The neural network is capable of analyzing much more complex patterns within

the data. It makes use of the additional information provided by each feature layer thus improving accuracy.

It may be surprising that in some cases, 100% accuracy was achieved. It must be remembered that this experiment is limited by a relatively small sample size. It is certainly possible to achieve a 100% accuracy rating over 20 samples. The best test for the accuracy of each model using the samples available is in the “Other” category. Because the total sample subset is over 8000 samples and each new testing group is chosen randomly, the models are tested against a much greater variety of samples. This makes the “Other” category a much more meaningful test of accuracy.

One of the challenges of using such a small sample set is the danger of false correlations. This is a possibility in both approaches. The rule-based comparison model relies on the average of the “Good Barks” subset. A small sample set may yield averages that are not really representative of all possible “Good Barks” that the dog might produce. In neural network training, reliance on a small sample set can result in overfitting. The neural network may become reliant on patterns that are not intrinsic to the real set but are merely coincidence. This is known as sampling noise. Adding samples with noise into the sample pool can help reduce this effect. The dropout layer in the neural network that was tested also helps reduce this effect. Still, there is no way to be sure that either of the classification methods would maintain their high accuracy unless more samples were introduced.

CHAPTER 4

CONCLUSION

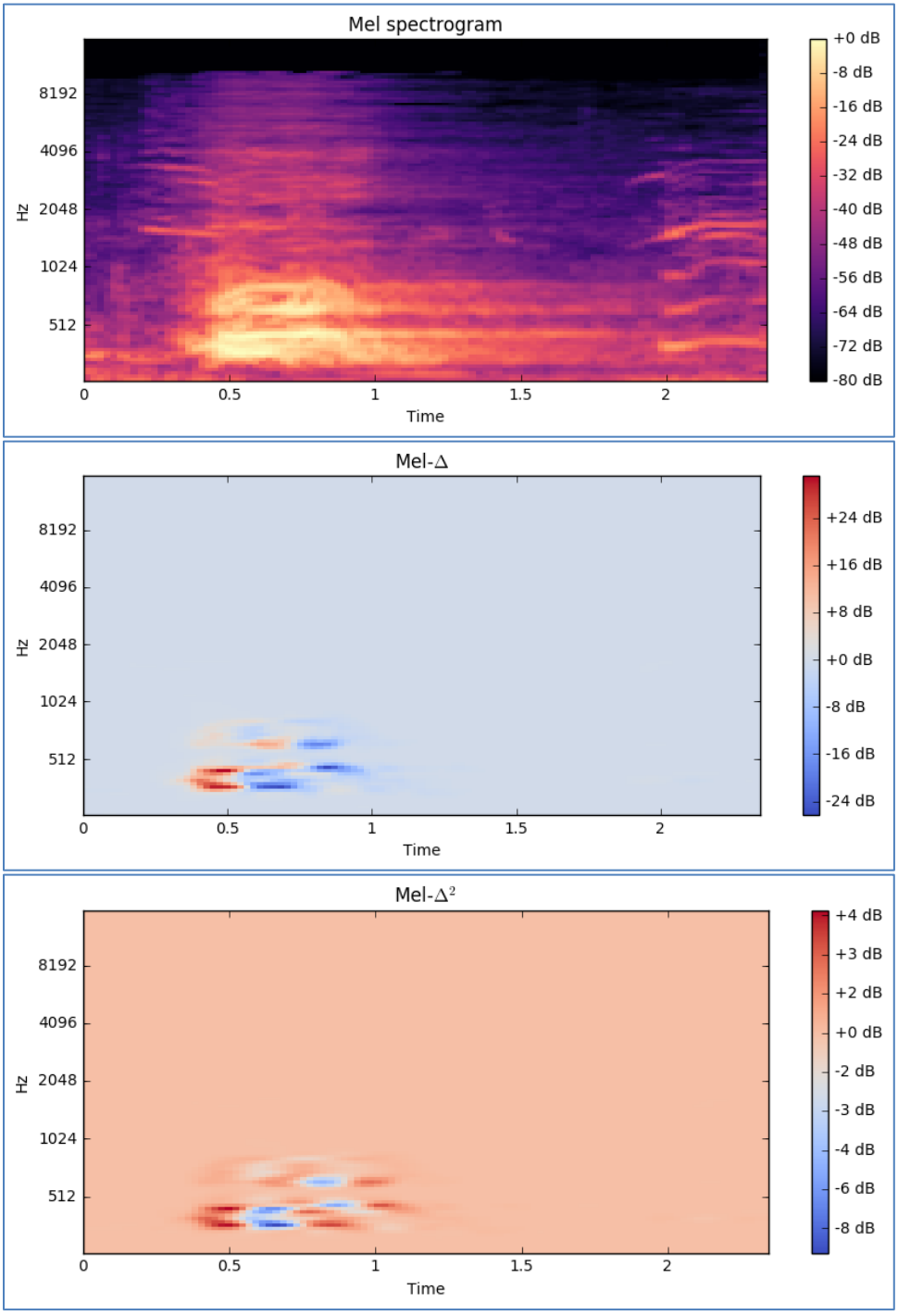
Only one neural network implementation was tested. There are undoubtedly many ways of improving on the network model. Adding new layers could further improve accuracy. Alternatively, reducing the number of layers may improve computational efficiency without compromising too much accuracy. Using a different training method could improve the time to convergence or allow the network to find a better minima. It may be worthwhile for future research to solve the problem of the full feature comparison classifier by introducing more complexity into the algorithm. Recording more samples (especially of "Good Barks") and retesting the models might also yield some interesting results.

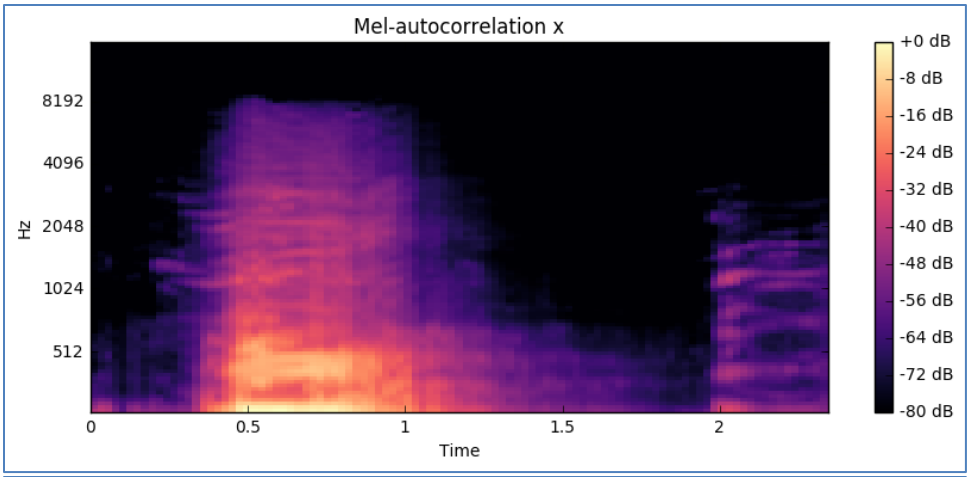
In this paper, it was demonstrated that neural networks have the potential to outperform conventional rule-based algorithms even in cases where the sample pool is small. Neural networks do not require any human decision process once implemented and can be easily retrained with new samples at any time to improve accuracy. Neural networks do, however, require significant memory resources to implement. They also require more computing power than rule-based methods. Although the Raspberry Pi is able to accommodate these requirements, other embedded systems may not. Even though only one possible rule-based algorithm was explored, the accuracy of the convolutional neural network was high enough that the advantages of the neural network approach would most likely outweigh the advantages of any rule-based approach for the application investigated.

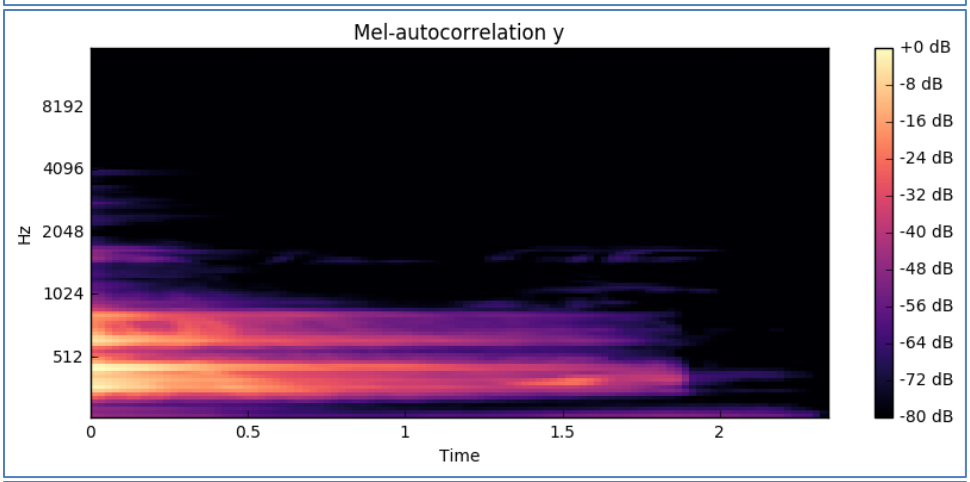
APPENDIX A

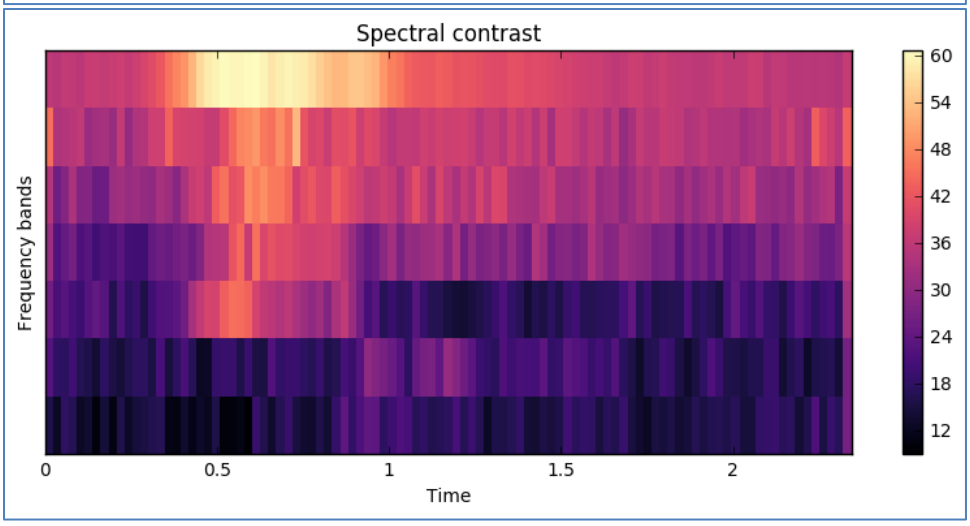
SAMPLE SET FEATURE COMPARISON

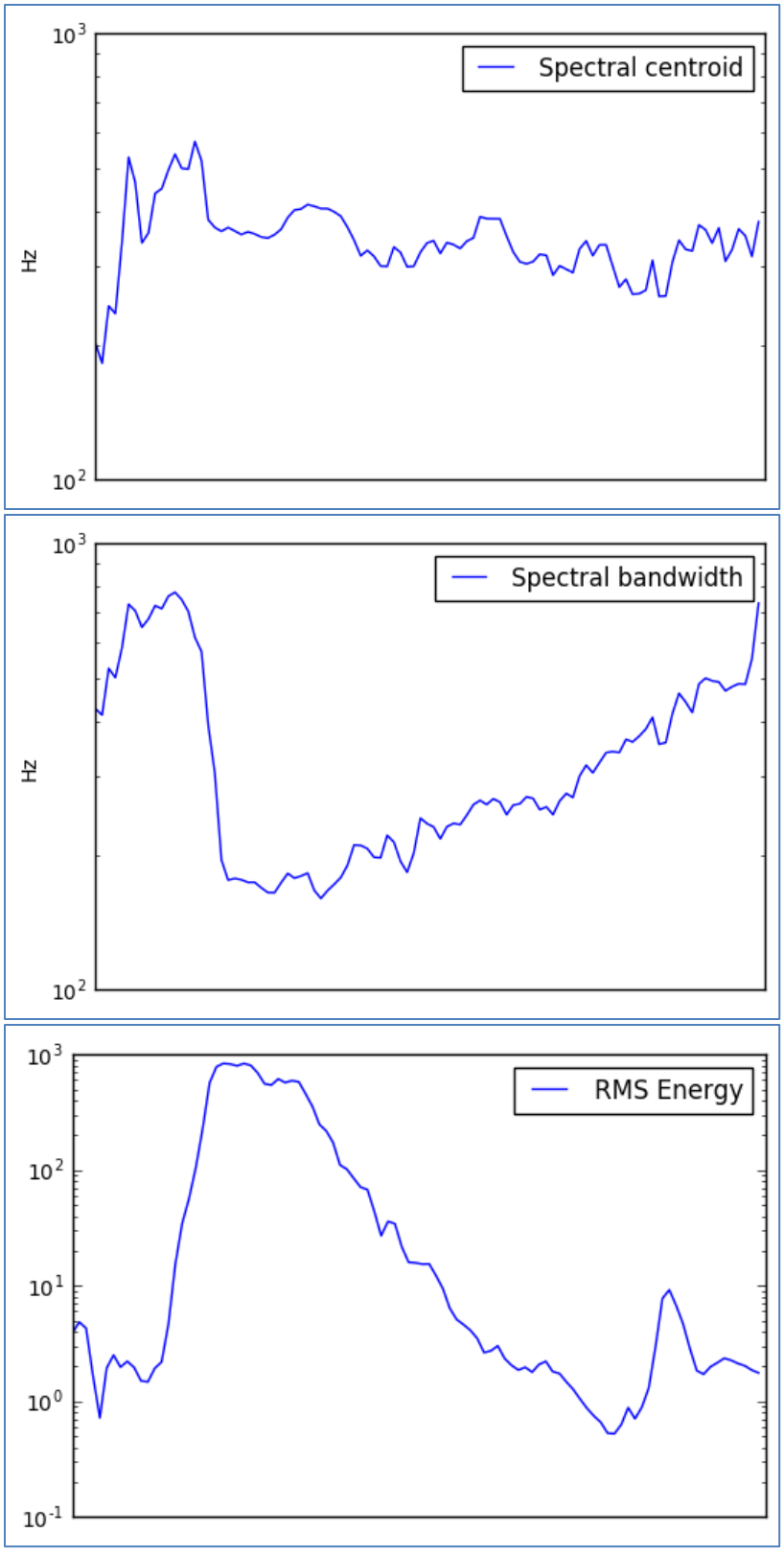
“GOOD BARK” FEATURE AVERAGES



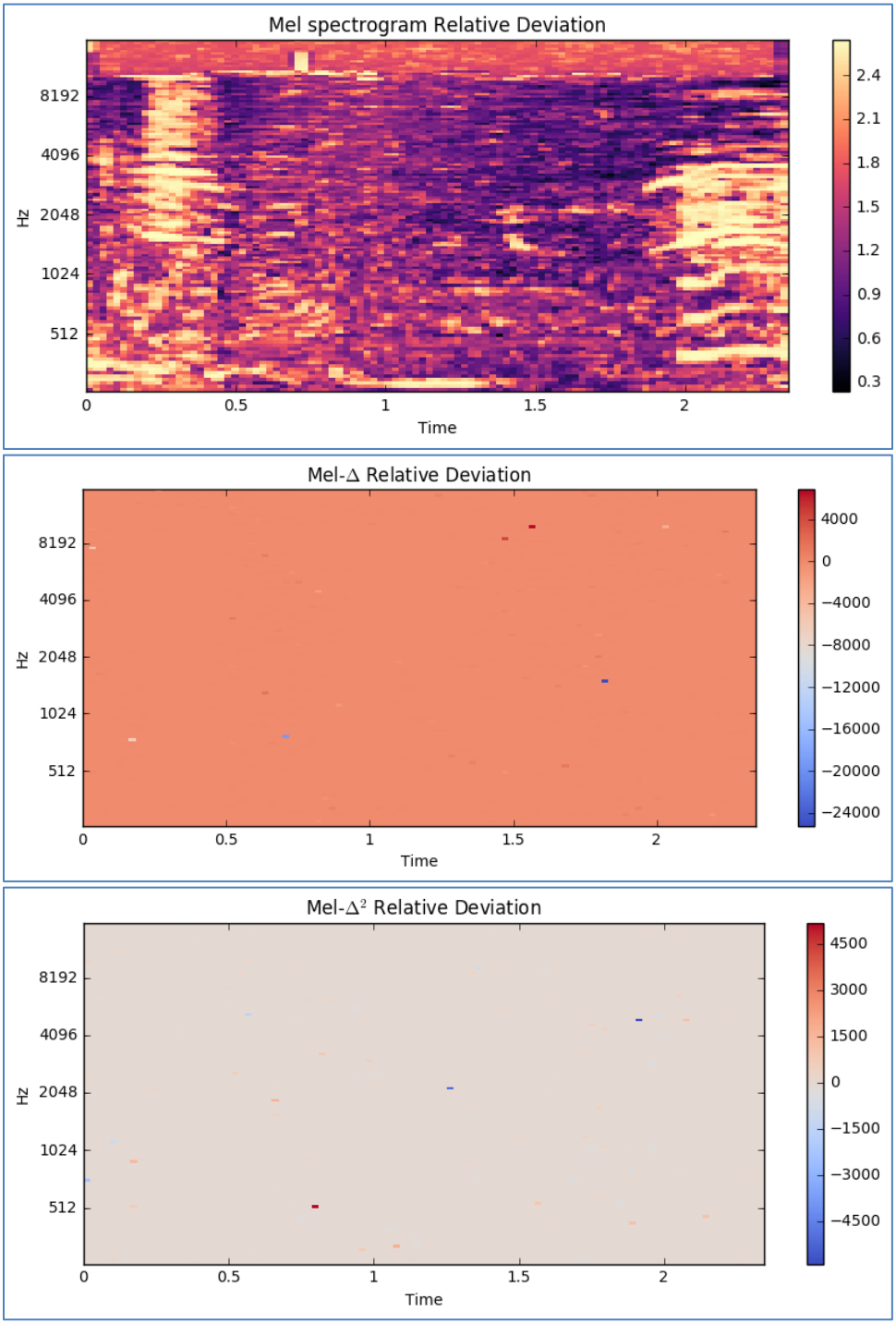

Mel-autocorrelation x
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
+0 dB
-8 dB
-16 dB
-24 dB
-32 dB
-40 dB
-48 dB
-56 dB
-64 dB
-72 dB
-80 dB

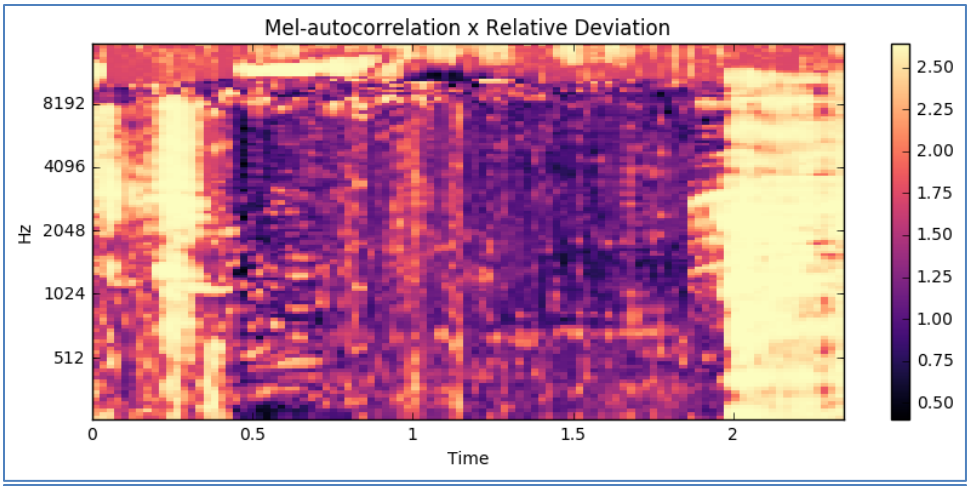

Mel-autocorrelation y
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
+0 dB
-8 dB
-16 dB
-24 dB
-32 dB
-40 dB
-48 dB
-56 dB
-64 dB
-72 dB
-80 dB

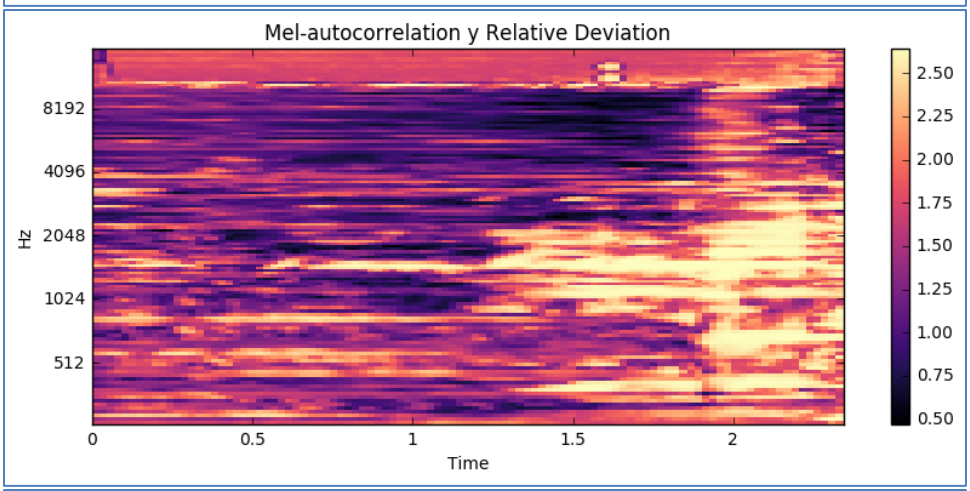

Spectral contrast
Frequency bands
0
0.5
1
1.5
2
Time
60
54
48
42
36
30
24
18
12

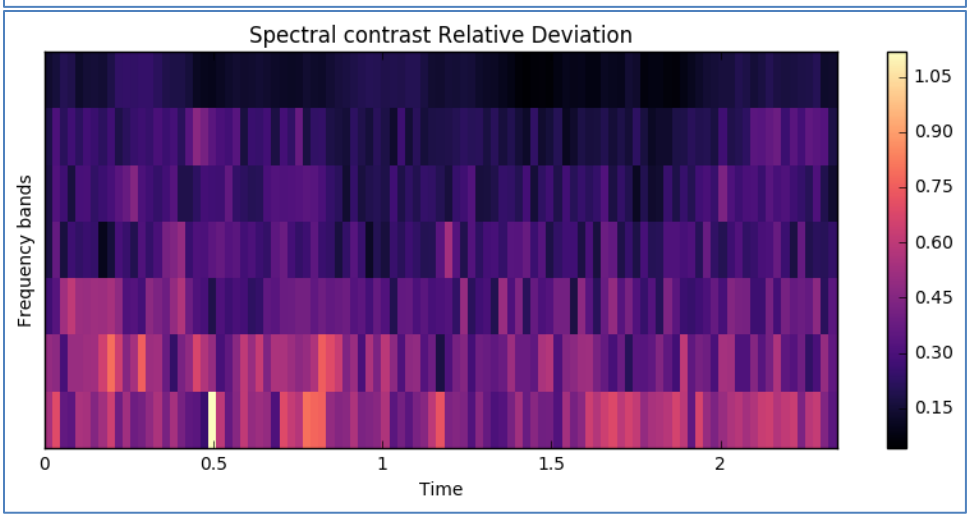

Spectral centroid
Hz
10^3
10^2
Spectral bandwidth
Hz
10^3
10^2
RMS Energy
10^3
10^2
10^1
10^0
10^{-1}

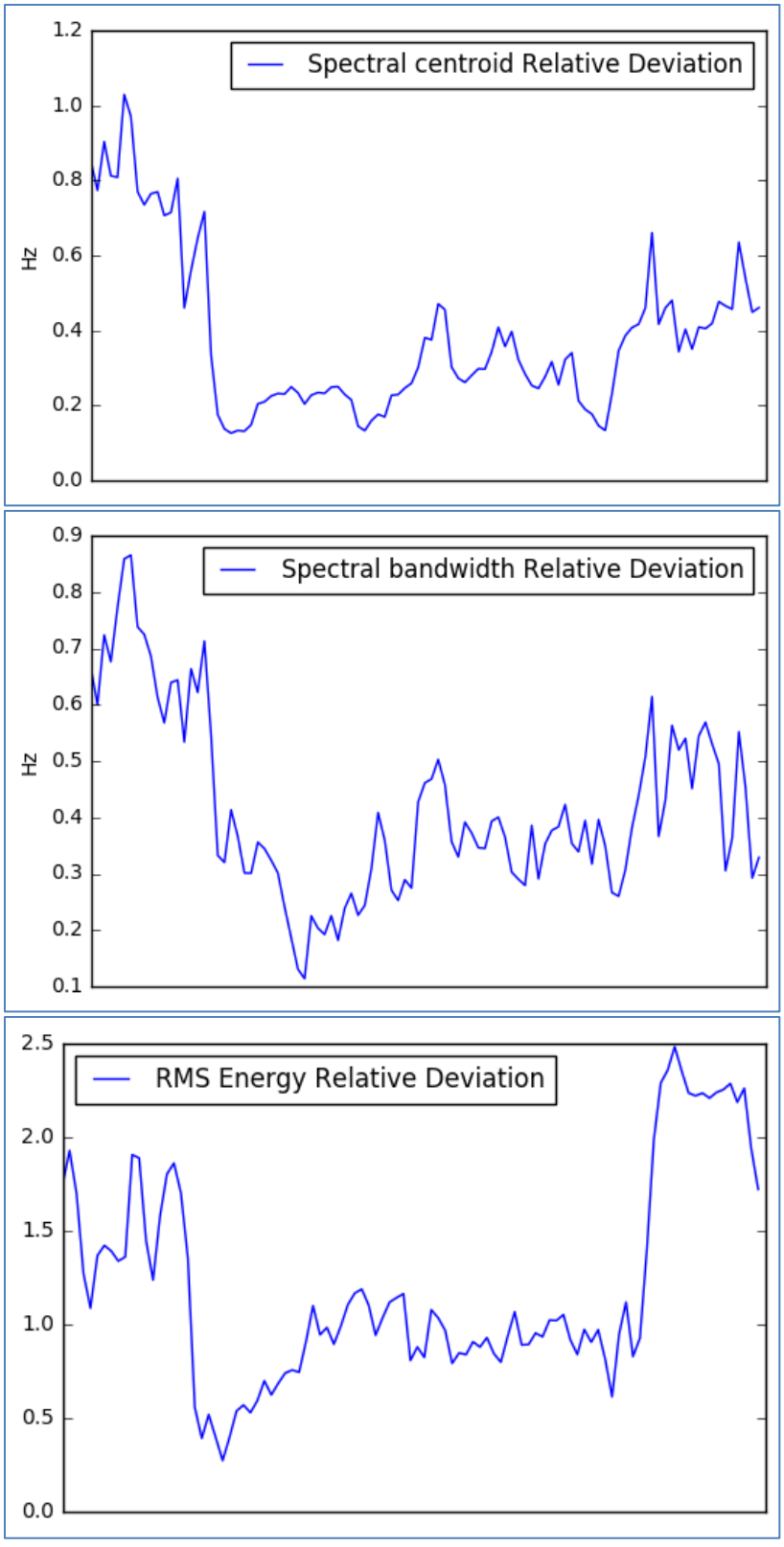
“GOOD BARKS” RELATIVE DEVIATION



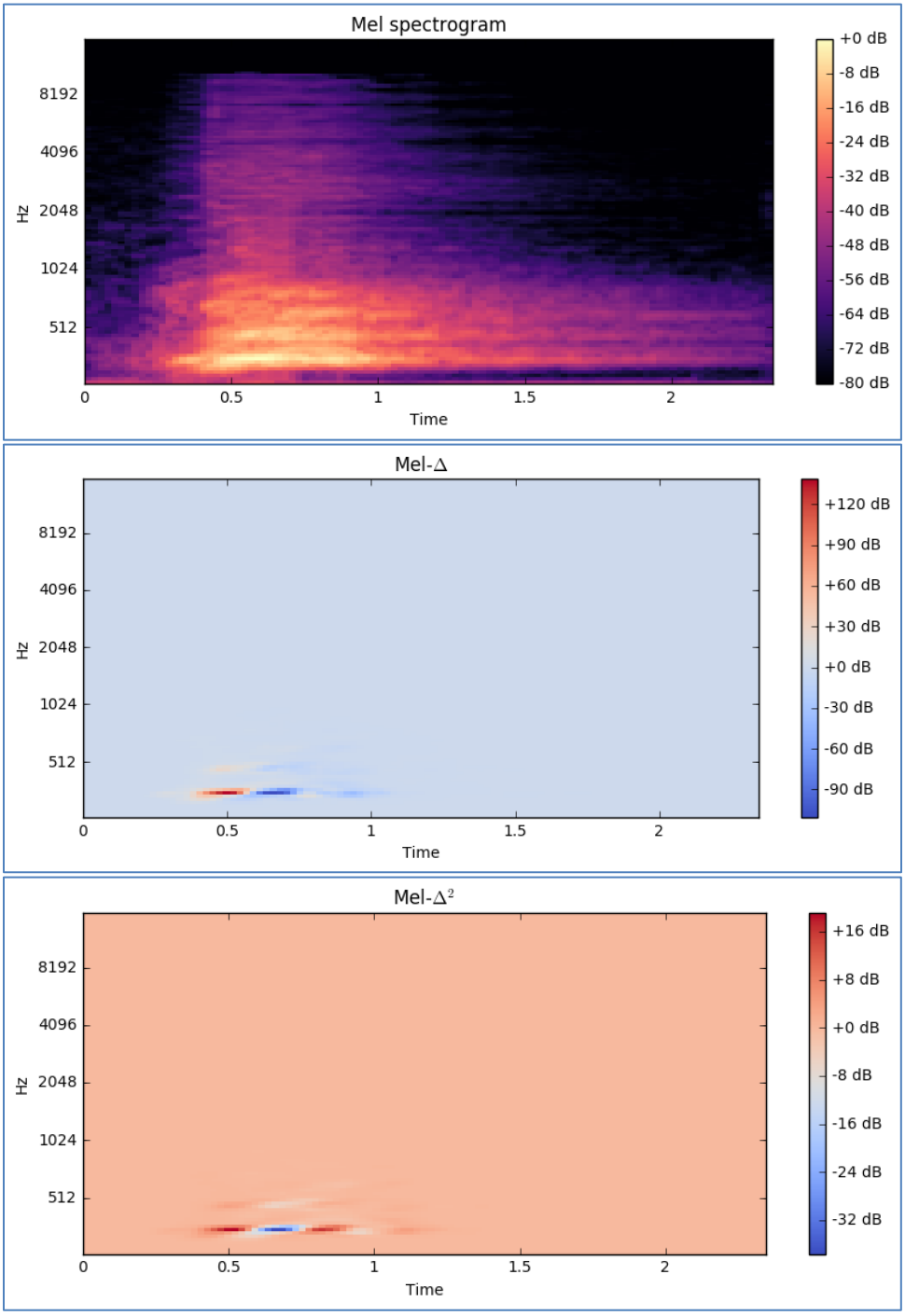

Mel-autocorrelation x Relative Deviation
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50

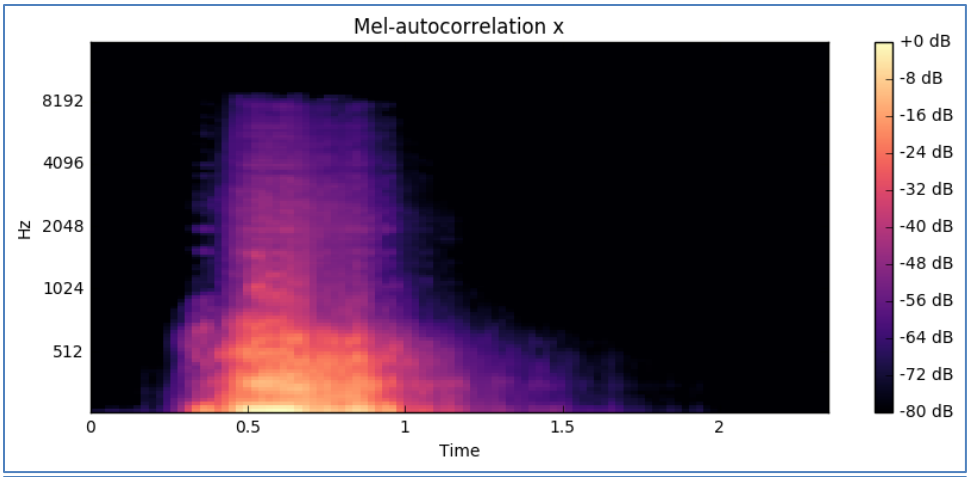

Mel-autocorrelation y Relative Deviation
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50

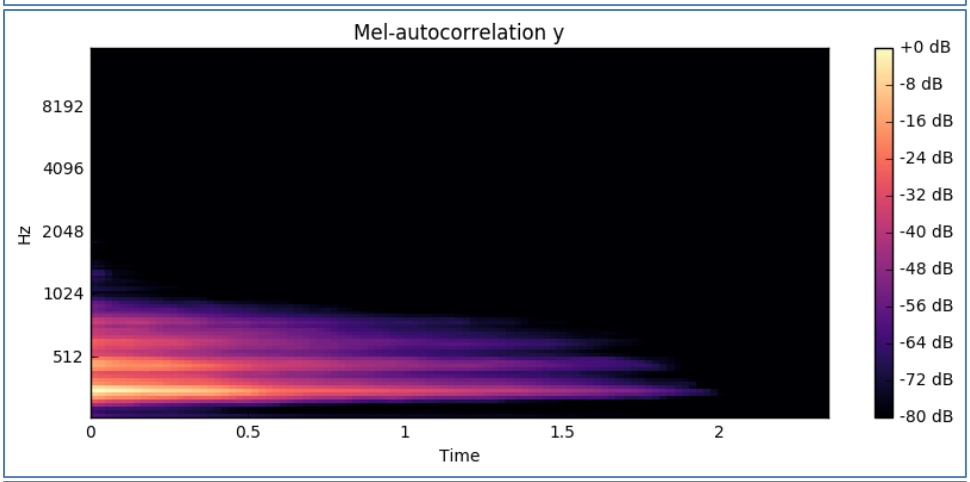

Spectral contrast Relative Deviation
Frequency bands
0
0.5
1
1.5
2
Time
1.05
0.90
0.75
0.60
0.45
0.30
0.15

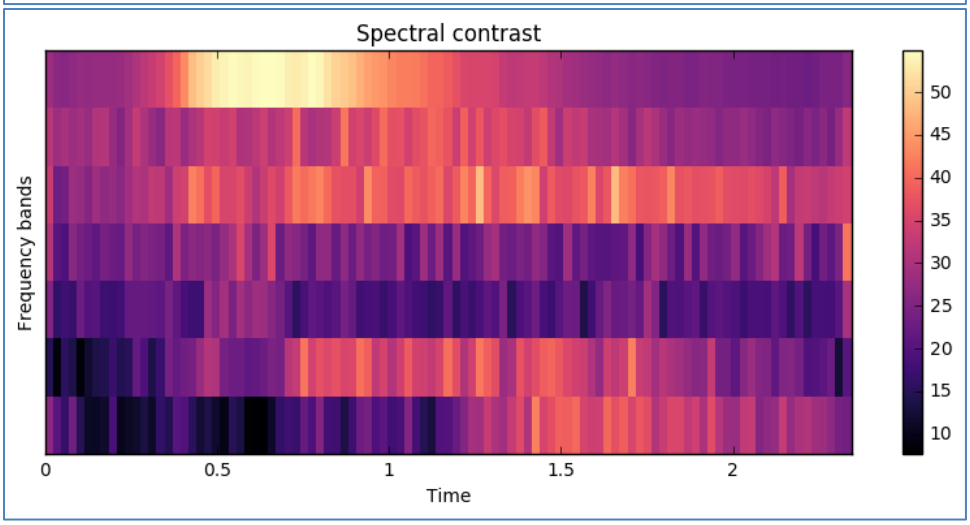

Spectral centroid Relative Deviation
Hz
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Spectral bandwidth Relative Deviation
Hz
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
RMS Energy Relative Deviation
2.5
2.0
1.5
1.0
0.5
0.0

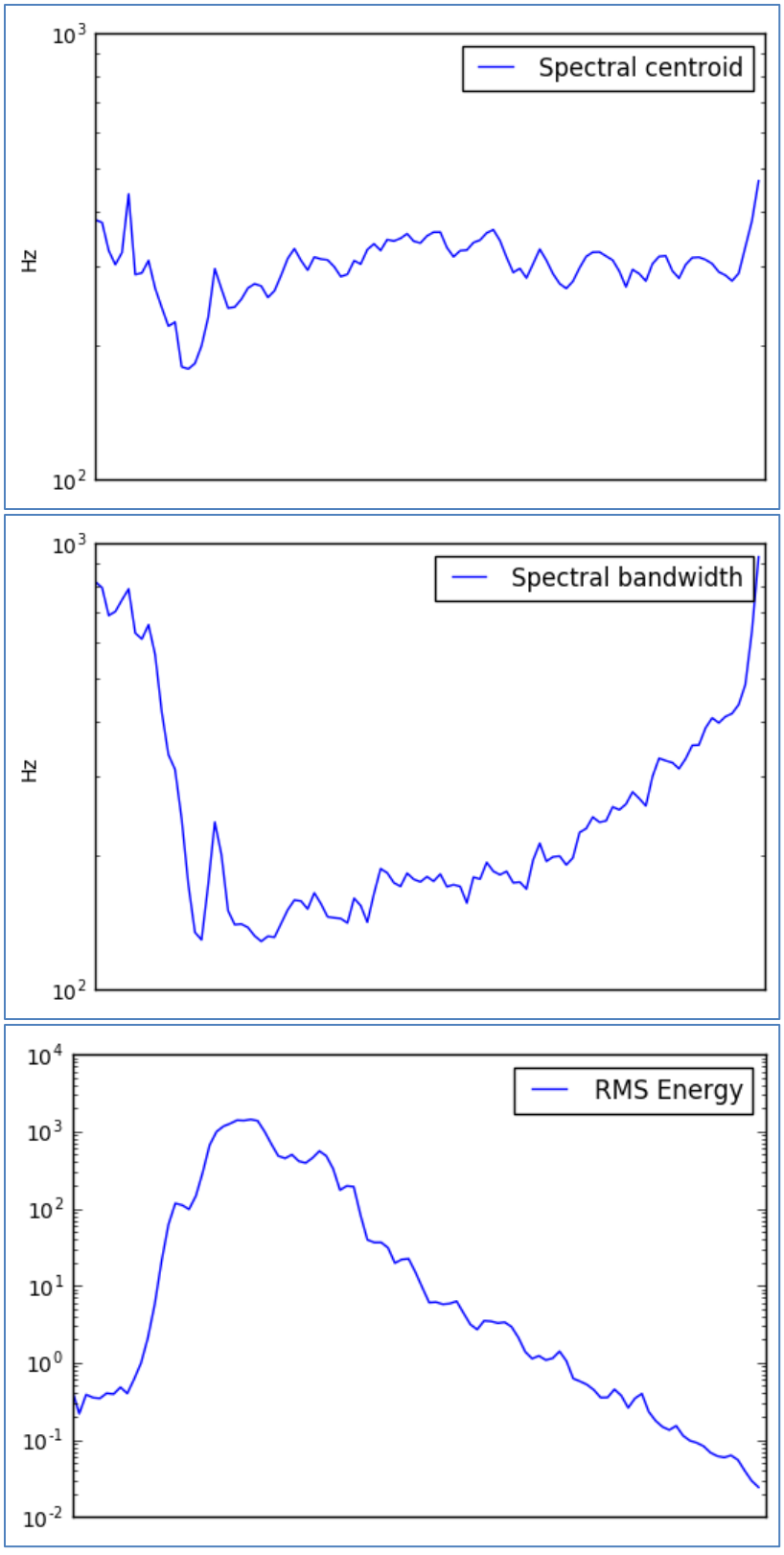
“BAD BARKS” FEATURE AVERAGES



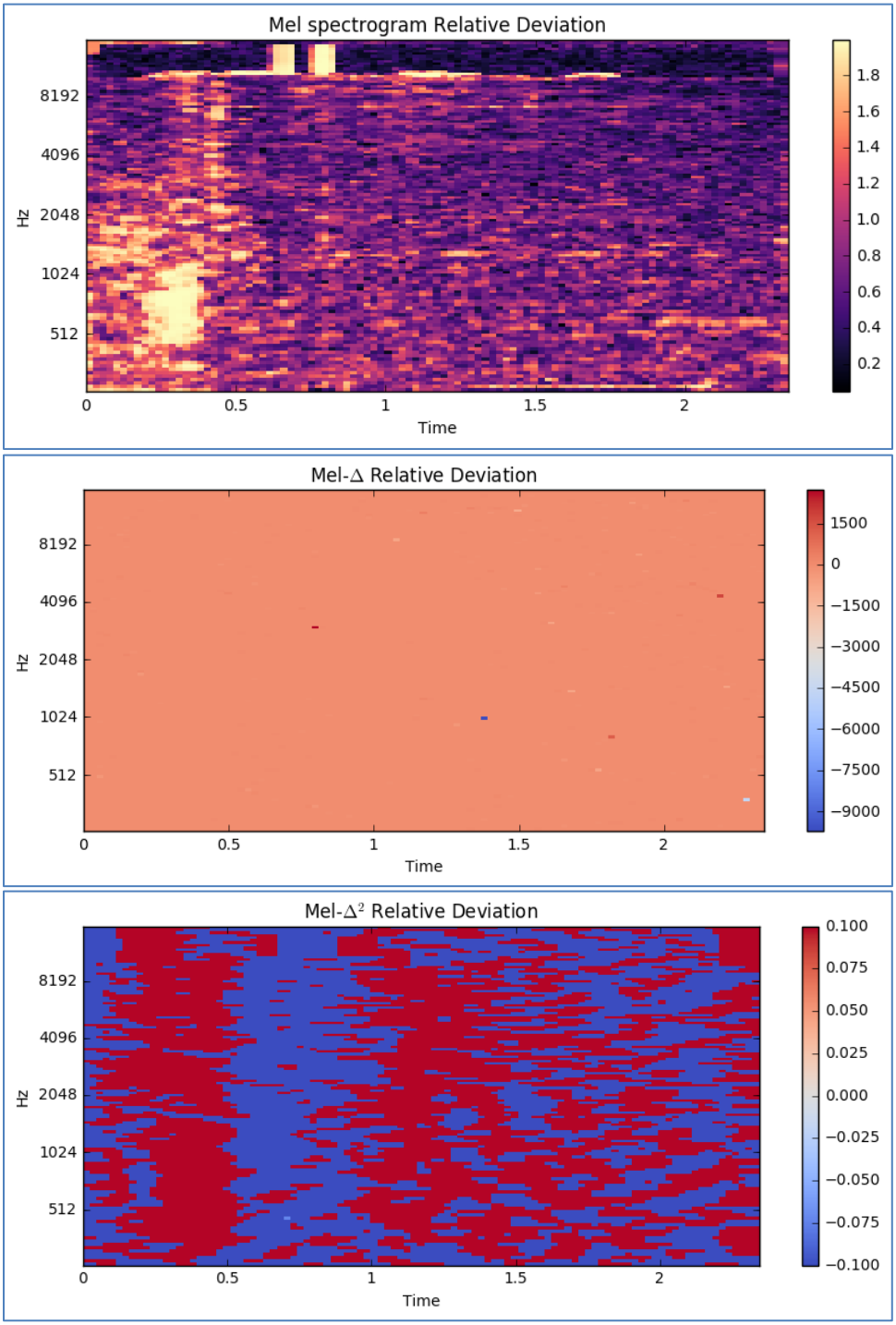

Mel-autocorrelation x
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
+0 dB
-8 dB
-16 dB
-24 dB
-32 dB
-40 dB
-48 dB
-56 dB
-64 dB
-72 dB
-80 dB

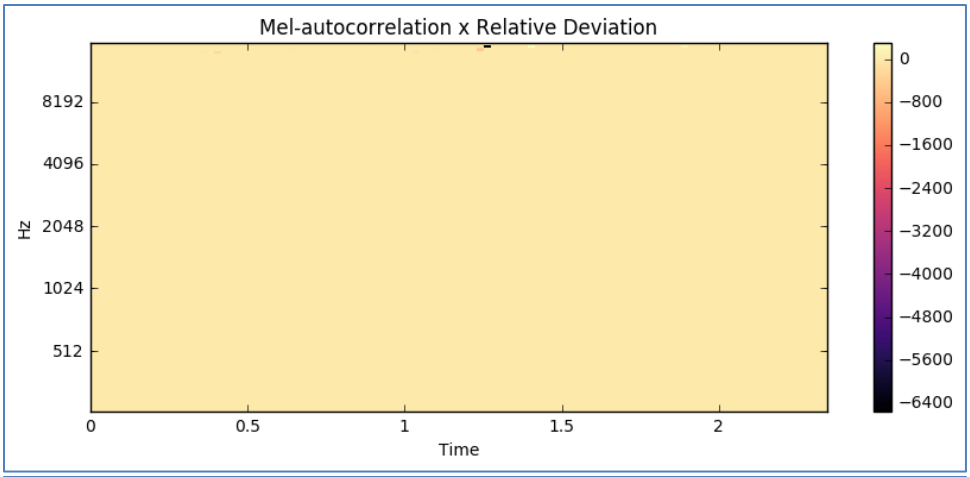

Mel-autocorrelation y
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
+0 dB
-8 dB
-16 dB
-24 dB
-32 dB
-40 dB
-48 dB
-56 dB
-64 dB
-72 dB
-80 dB

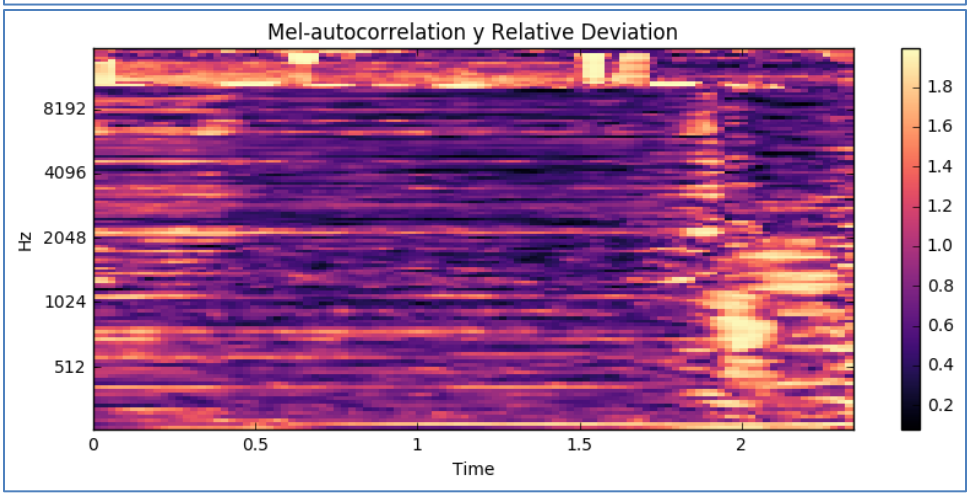

Spectral contrast
Frequency bands
0
0.5
1
1.5
2
Time
50
45
40
35
30
25
20
15
10

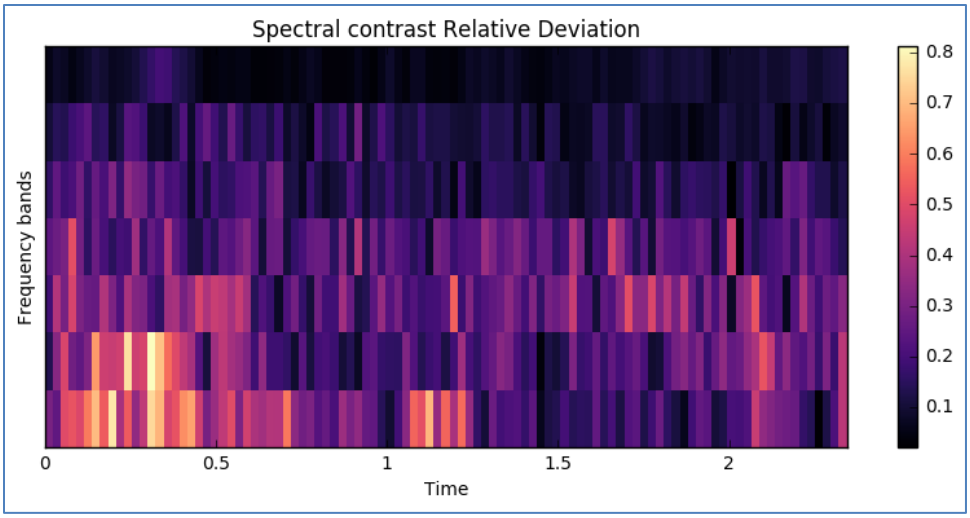

Spectral centroid
Hz
10^3
10^2
Spectral bandwidth
Hz
10^3
10^2
RMS Energy
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}

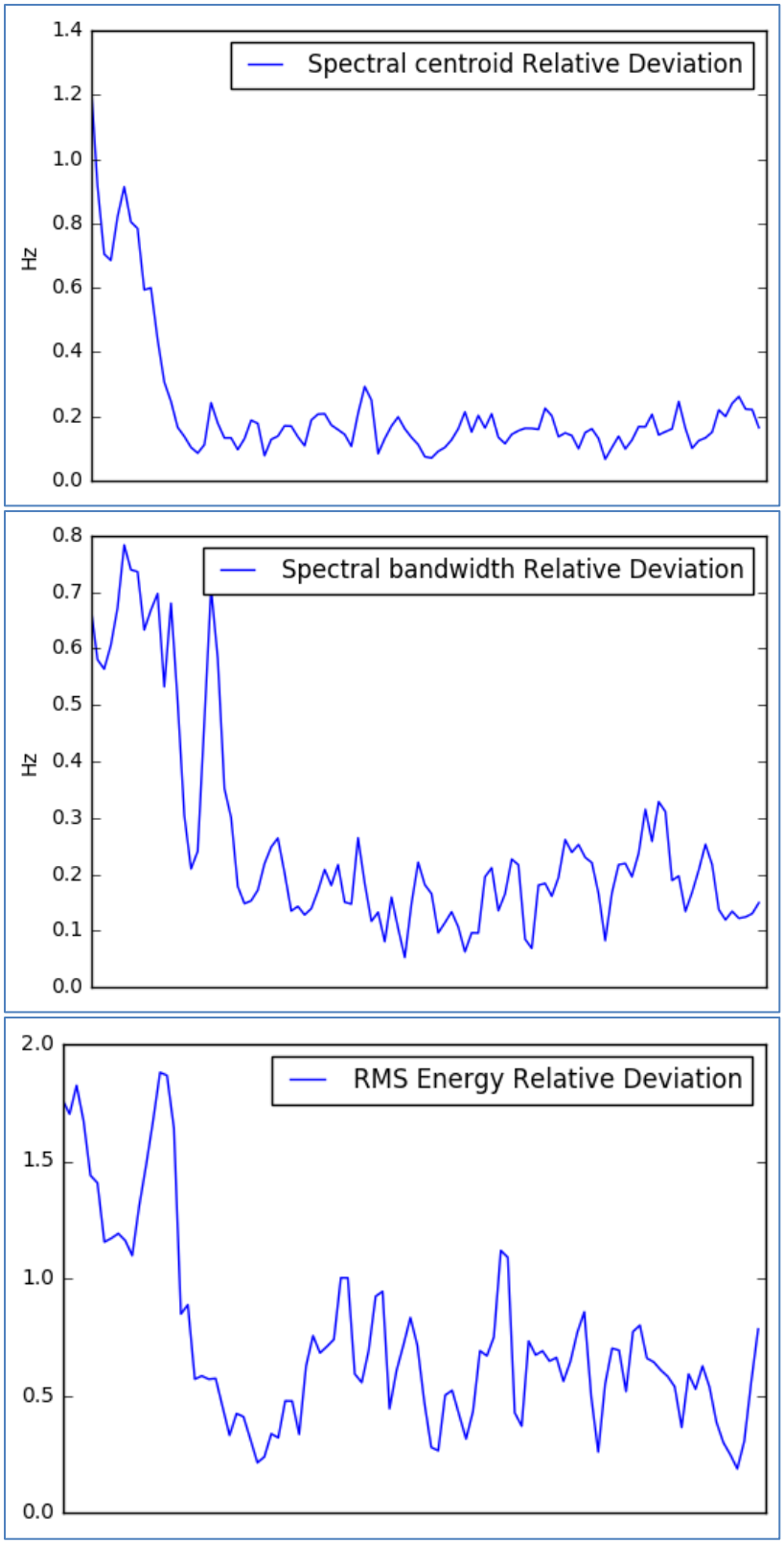
“BAD BARKS” RELATIVE DEVIATION



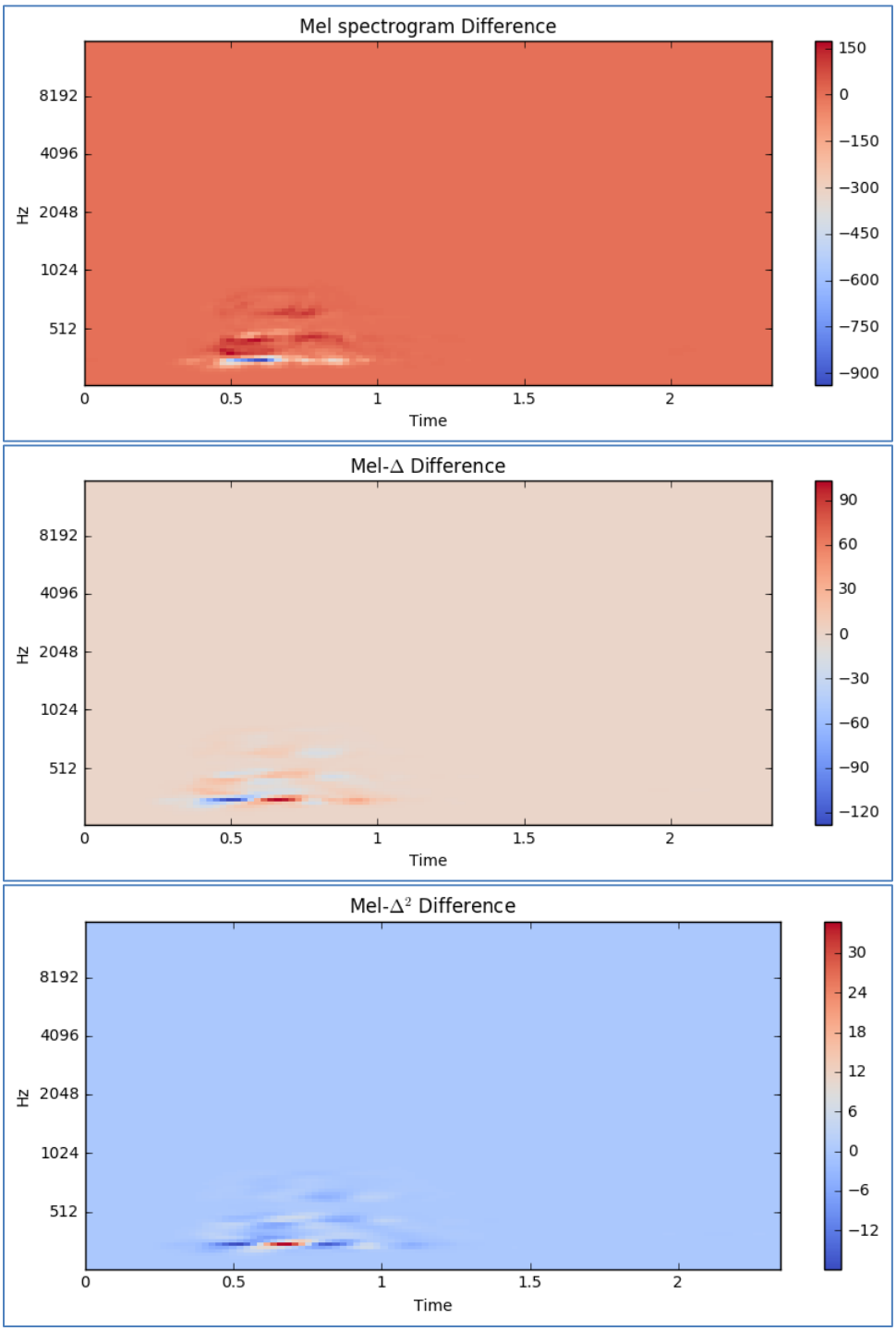

Mel-autocorrelation x Relative Deviation
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
0
−800
−1600
−2400
−3200
−4000
−4800
−5600
−6400

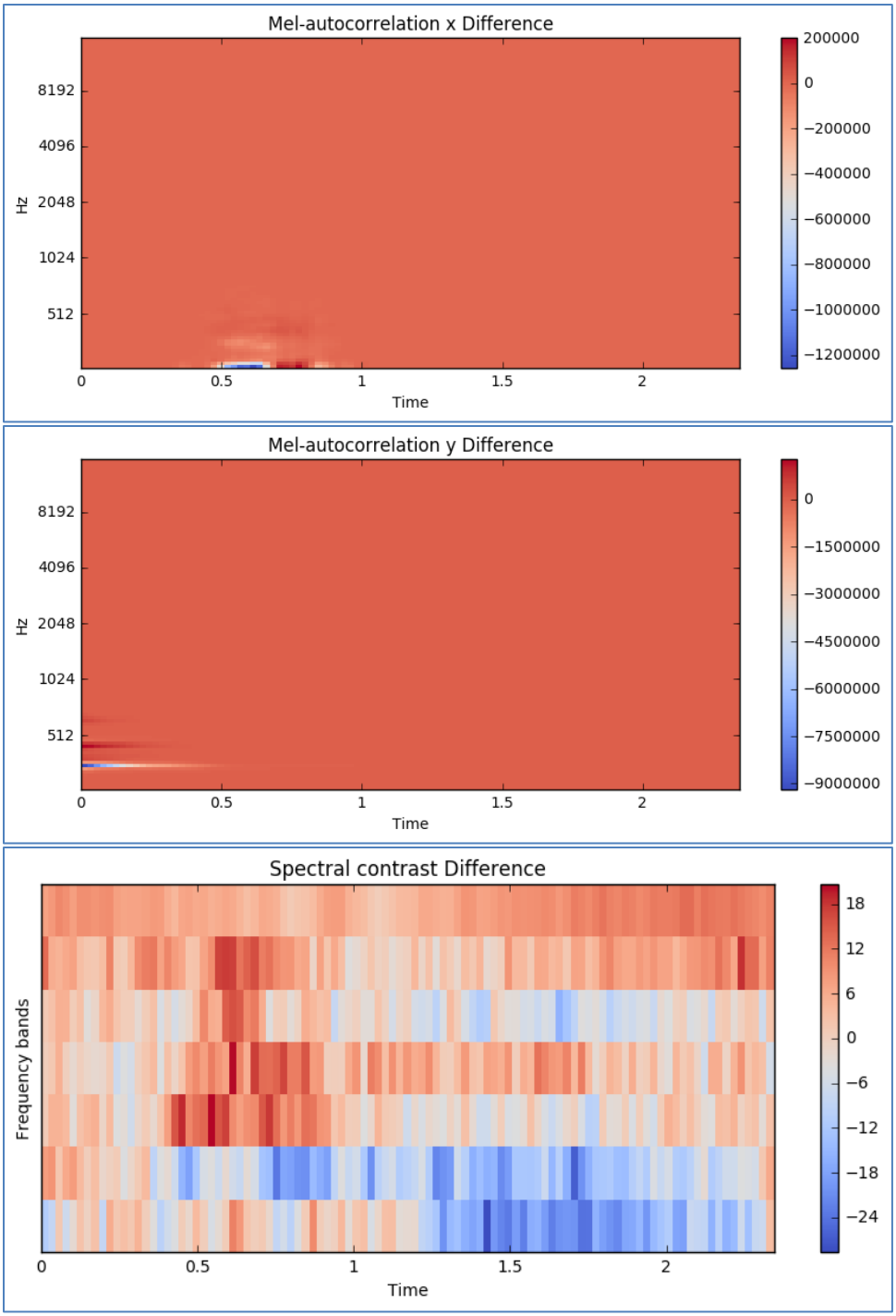

Mel-autocorrelation y Relative Deviation
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2

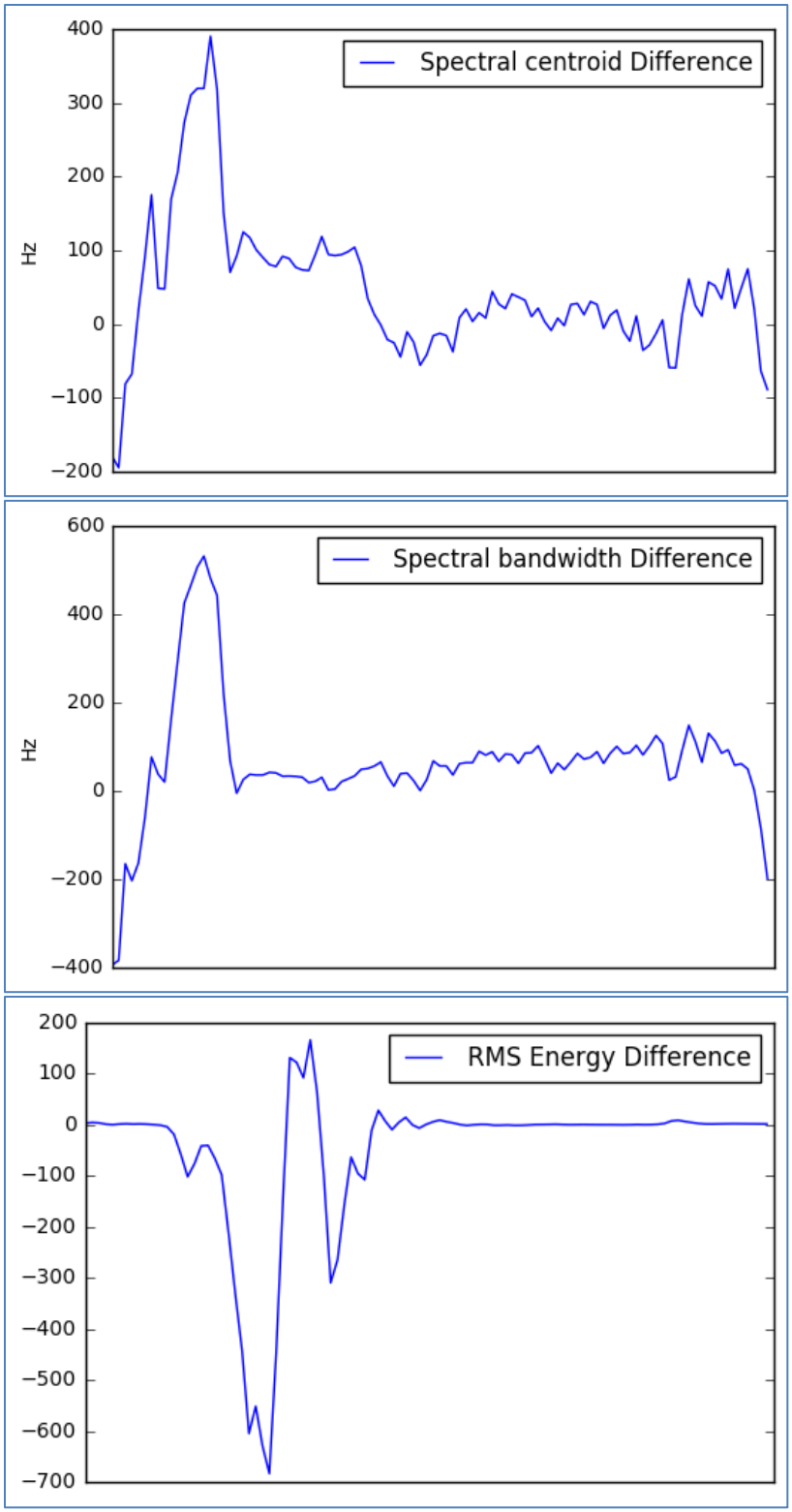

Spectral contrast Relative Deviation
Frequency bands
0
0.5
1
1.5
2
Time
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1


Spectral centroid Relative Deviation
Hz
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Spectral bandwidth Relative Deviation
Hz
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
RMS Energy Relative Deviation
2.0
1.5
1.0
0.5
0.0

DIFFERENCE BETWEEN FEATURES FROM
“GOOD BARKS” AND “BAD BARKS”




Mel-autocorrelation x Difference
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
200000
0
−200000
−400000
−600000
−800000
−1000000
−1200000
Mel-autocorrelation y Difference
8192
4096
2048
1024
512
Hz
0
0.5
1
1.5
2
Time
0
−1500000
−3000000
−4500000
−6000000
−7500000
−9000000
Spectral contrast Difference
Frequency bands
0
0.5
1
1.5
2
Time
18
12
6
0
−6
−12
−18
−24


Spectral centroid Difference
Hz
400
300
200
100
0
−100
−200
Spectral bandwidth Difference
Hz
600
400
200
0
−200
−400
RMS Energy Difference
200
100
0
−100
−200
−300
−400
−500
−600
−700

APPENDIX B

CLASSIFICATION TEST RESULTS COMPARISON

Test Run 1	"Good Barks" correctly classified as 1	"Bad Barks" correctly classified as 0	Other Sounds correctly classified as 0
Frequency Comparison	100%	100%	94%
Spectrogram Comparison	100%	100%	98%
Full Feature Comparison	100%	60%	99%
Convolutional Neural Network	100%	100%	99%

Test Run 2	"Good Barks" correctly classified as 1	"Bad Barks" correctly classified as 0	Other Sounds correctly classified as 0
Frequency Comparison	100%	100%	96%
Spectrogram Comparison	100%	100%	99%
Full Feature Comparison	100%	50%	99%
Convolutional Neural Network	100%	100%	99%

Test Run 3	"Good Barks" correctly classified as 1	"Bad Barks" correctly classified as 0	Other Sounds correctly classified as 0
Frequency Comparison	100%	100%	95%
Spectrogram Comparison	100%	100%	99%
Full Feature Comparison	100%	65%	99%
Convolutional Neural Network	100%	100%	100%

Test Run 4	"Good Barks" correctly classified as 1	"Bad Barks" correctly classified as 0	Other Sounds correctly classified as 0
Frequency Comparison	100%	100%	94%
Spectrogram Comparison	100%	100%	95%
Full Feature Comparison	100%	70%	100%
Convolutional Neural Network	100%	100%	100%

Test Run 5	"Good Barks" correctly classified as 1	"Bad Barks" correctly classified as 0	Other Sounds correctly classified as 0
Frequency Comparison	100%	100%	96%
Spectrogram Comparison	100%	100%	99%
Full Feature Comparison	100%	55%	99%
Convolutional Neural Network	100%	100%	100%

Test Run 6	"Good Barks" correctly classified as 1	"Bad Barks" correctly classified as 0	Other Sounds correctly classified as 0
Frequency Comparison	100%	100%	97%
Spectrogram Comparison	100%	100%	99%
Full Feature Comparison	100%	60%	99%
Convolutional Neural Network	100%	100%	100%

REFERENCES

[1] “Raspberry Pi 3 Model B,” *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 05-May-2017].

[2] M. Hagan, H. Demuth, and M. Beale, *Neural Network Design*. PWS Publishing Company, 1996.

[3] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall; 2nd edition, 1998.

[4] S. Ruder, “*An Overview of Gradient Descent Optimization Algorithms*,” [1609.04747] An overview of gradient descent optimization algorithms, 15-Sep-2016. [Online]. Available: https://arxiv.org/abs/1609.04747. [Accessed: 05-May-2017].

[5] J. Salamon, C. Jacoby and J. P. Bello, "*A Dataset and Taxonomy for Urban Sound Research*", 22nd ACM International Conference on Multimedia, Orlando USA, Nov. 2014.

[6] A. Saeed, “*Urban Sound Classification*,” Aaqib Saeed. [Online]. Available: http://aqibsaeed.github.io/2016-09-03-urban-sound-classification-part-1/. [Accessed: 05-May-2017].

[7] D. Li, I. Sethi, N. Dimitrova, and T. McGee, “*Classification of general audio data for content-based retrieval”*, Elsevier Science, 2000

[8] A.Mohamed, *“Deep Neural Network acoustic models for ASR”*, University of Toronto, 2014.

[9] M. Karbasi, S. M. Ahadi, and M. Bahmanian, "*Environmental sound classification using spectral dynamic features*", 2011 8th International Conference on Information, Communications & Signal Processing, 2011.

[10] S. Chachada and C.-C. J. Kuo, "*Environmental sound recognition: A survey,*" 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, 2013.

[11] T. Makinen, S. Kiranyaz, and M. Gabbouj, "*Content-based audio classification using collective network of binary classifiers*," 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS), 2011.

[12] Wang, A., "*An Industrial-Strength Audio Search Algorithm,"* Proceedings of the 4th International Conference on Music Information Retrieval, 2003.

[13] "TensorFlow," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/. [Accessed: 5-May-2017].

[14] "LibROSA," *LibROSA — librosa 0.5.1 documentation*. [Online]. Available: https://librosa.github.io/librosa/. [Accessed: 5-May-2017].

BIOGRAPHICAL INFORMATION

Bradley Wabbersen is completing his Honors Bachelor of Science in Electrical Engineering at the University of Texas at Arlington. After graduation, he will begin work at Harris Corporation in Palm Bay, Florida as an electrical engineer. During the Summer of 2016, he completed an internship with Harris, focusing largely on Field Programmable Gate Array (FPGA) firmware design. His academic career has primarily been on the field of embedded systems and microprocessors. He also has a strong interest in machine learning and intelligent algorithms. Over the past four years, he has worked on a variety of projects ranging from Printed Wire Board (PWB) fabrication to mesh network design. In the future, Bradley plans to continue his education by pursuing a master's degree in Electrical Engineering.