Deep Sample: A Study of Audio Segmentation

Andrew Vincent Moore, Hue Truong, and Alex Reno

Department of Computer Science, Quinsigamond Community College,
01606 Worcester, MA, United States

Correspondence should be addressed to:

Andrew Vincent Moore	*amoore15@qmail.qcc.edu*
Hue Truong	*htylertruong@gmail.com*
Alexander Reno	*areno6@qmail.qcc.edu*
Professor Hao Loi	*hloi@qcc.mass.edu*

Abstract

Audio segmentation is a preprocessing step in signal analysis that breaks an audio signal into individual components for use in further analysis. Modern audio analysis uses a variety of segmentation algorithms in combination with machine learning techniques to accomplish this. Deep Sample is a two stage study seeking to evaluate both the audio segmentation algorithms themselves and their effectiveness when combined with machine learning techniques. The first stage examines the efficacy and efficiency of five popular audio segmentation algorithms: Cooley-Tukey Fast Fourier Transform, Zero Crossing, Spectrum Flux, Spectrum Centroid, and Real Cepstrum. The efficacy is determined by analyzing results of an audio segmentation algorithm given known data. The efficiency is determined by comparing execution times for each algorithm given the known data set. The second stage of the study utilizes a neural network Deep Sample. Deep Sample performs a learning vector quantization algorithm to predict the genre of an audio signal based on the audio segmentation algorithms implemented in the first stage. The focus of this part of the study is the efficacy and efficiency of Deep Sample, each of which is determined by comparing results with each audio segmentation algorithm. The basis of this research can be found in Fawad Hussain's *Optimized Audio Classification and Segmentation Algorithm by Using Ensemble Methods*[1].

1. Introduction

Audio segmentation is a preprocessing step in audio analysis that separates different parts of the audio signal [3]. Modern audio analysis uses a variety of segmentation algorithms in combination with machine learning techniques. Deep Sample is a two stage study seeking to evaluate both the audio segmentation algorithms themselves as well as the effectiveness of each algorithm when combined with machine learning techniques.

The first stage focuses on popular audio segmentation algorithms, analyzing both their efficacy and efficiency. The algorithms chosen in this study are the Cooley-Tukey Fast Fourier Transform, Zero Crossing, Spectral Flux, Spectrum Centroid, and Real Cepstrum algorithms.

The efficacy of an algorithm is measured by comparing the output of said algorithm with a known dataset. Efficiency is a measure of the amount of time the algorithm takes to complete when compared to a run against a standard sample input.

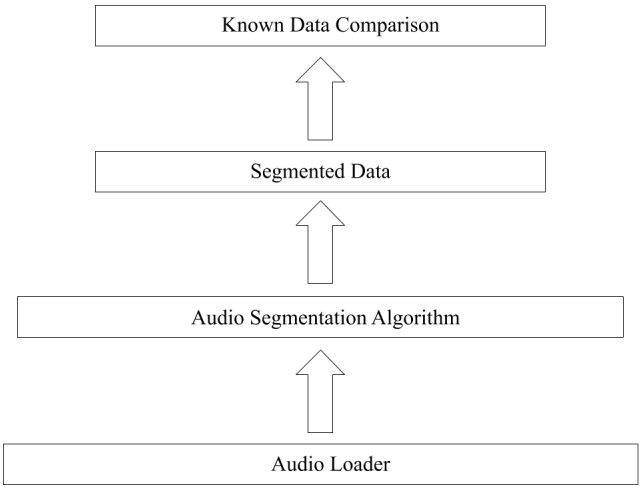

Figure 1: Block Diagram of Audio Segmentation Algorithm.

The second stage feeds the output of the first stage into Deep Sample, a neural network performing a learning vector quantization algorithm to determine the genre of the given audio signal.

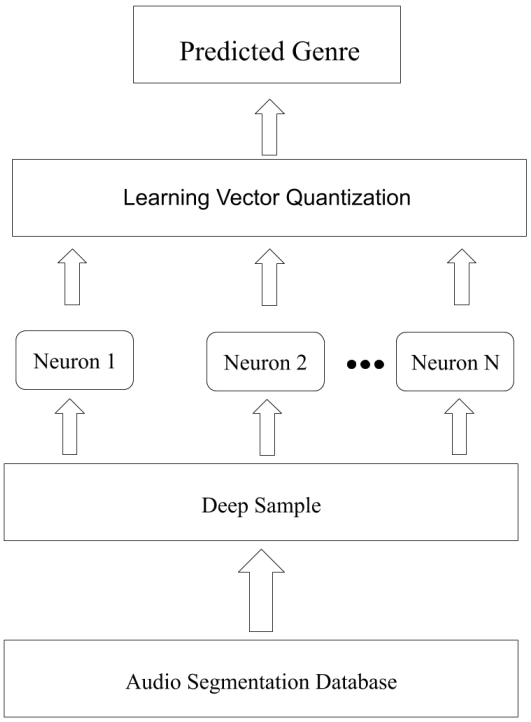

Figure 2: Block Diagram of Deep Sample

The results of Deep Sample can be used in a wide variety of audio classification projects. Identifying specific musical pieces, musical genres, differentiating between spoken and instrumental pieces, and other applications are possible after the signal has been preprocessed and analyzed by Deep Sample.

2. Background

2.1 Cooley-Tukey Fast Fourier Transform

The Cooley-Tukey Fourier Transform is a function that decomposes a signal into its component frequencies. Whereas a raw signal is usually defined by its amplitude in respect to time, the fourier transform defines the signal by its frequency in respect to time. The fourier transform extends from the Fourier series, when the limit of the periodicity approaches infinity. The fourier

transform is most commonly used to analyze frequencies of a signal, and is given by the equation:

$$F(x) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x\xi}dx \quad (1)\ [4]$$

2.2 Zero Crossing

The zero crossing rate is a measure of the noisiness of a signal [4]. This is calculated as a summation of the noise level across each frame in the audio file.

$$Z_t = \frac{1}{2}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])| \quad (2)\ [4]$$

Where Z_t is the current frame of the audio file, and the sign function is 1 for positive arguments and 0 for negative arguments. The element $x[n]$ is the time domain signal for frame t [4].

The results of this algorithm can be used to find the average noise level in a sample, and then to compare that noise level to a set of known samples to determine the genre. This process is described in detail in section 2.6.

2.3 Spectrum Flux

The spectral flux is the sum of the squared difference between the normalized magnitudes of successive spectral distributions [4].

$$F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2 \quad (3)\ [4]$$

Where F_t is the spectral flux of the signal, N is the total number of elements in the current frame, n is the current element, $N_t[n]$ is the spectral distribution at element n of frame t, and $N_{t\text{-}1}[n]$ is the spectral distribution at element n of frame t-1. A spectral distribution is represented by the Cooley-Tukey Fast Fourier Transform of the signal at each point in a frame. See section 2.1 for more details on the Cooley-Tukey Fast Fourier Transform.

The Spectrum Flux algorithm yields a single floating point value per channel, representing the spectral flux across the entire audio sample. This shows the amount of local spectral change in the sample [4], which is a representation of the change in signal strength. The Spectral Flux corresponds to the timbre of the audio signal, and can be leveraged for classification purposes by comparing to the timbre of known samples.

2.4 Spectrum Centroid

The spectral centroid is the center of gravity of the magnitude spectrum of the short-time Fourier transform [4].

$$C_t = \frac{\sum_{n=1}^{N} M_t[n]*n}{\sum_{n=1}^{N} M_t[n]} \qquad (4)\ [4]$$

Where C_t is the spectral centroid, N is the total number of elements, n is the current element, and $M_t[n]$ is the spectral distribution of element n of frame t. As with Spectrum Flux, the spectral distribution is represented by the Cooley-Tukey Fast Fourier Transform.

The Spectrum Centroid algorithm results in one floating-point value per channel, representing the spectral centroid across the entire audio sample. This value measures the spectral shape, which in turn corresponds to the "brightness" of the signal, with higher values corresponding to brighter textures with higher frequencies [4]. By comparing the spectral centroid value with known samples, audio can be categorized based on brightness.

2.5 Real Cepstrum

Real cepstrum is the result of a transformation of the Fourier transform. The real cepstrum is mathematically defined as:

$$C_r = F^{-1}\{log(|F\{f(t)\}|)\} \quad (5)[4]$$

Where F defines a Fourier transform, F^{-1} is the inverse Fourier transform, and $f(t)$ is the signal manipulated.

The real cepstrum defines the real parts of the fourier transform, as opposed to complex numbers. Cepstrum transforms in general are used to analyze periodicity within a frequency. Moreover, the real cepstrum can specifically be used to analyze the amplitude of a spectrum over time.

Though, the cepstrum must first be passed through a hamming window to taper off the data set at each end to zero, smoothing discontinuities of the sampled signal. The hamming window is defined as:

$$w(n) = 0.54 - 0.46cos(2\pi n/(M-1)) \quad (6)\ [4]$$

where n is the sample data, and M is the size of the window. The size of the window is determined by the signal's sample rate, frequency, and sample window.

2.6 Deep Sample

Deep Sample is an artificial neural network based on the work of Jason Brownlee [6]. The algorithm chosen for this project is Learning Vector Quantization (LVQ). This is a machine learning method that takes a vector of data and divides it into random codebooks, which it then uses as a training set to form a basis for comparing new data [6]. The main advantage of this approach is to decrease the size of the patterns being learned, allowing efficient analysis of large data sets.

The first step of LVQ is generating the code books. This is accomplished by taking a random range from the dataset, placing it in vector form, and calculating the best matching unit (BMU) between each vector.

The best matching unit is given by calculating the euclidean distance between each vector, represented by

$$d(x,y) = \sqrt{\sum_{n=0}^{N} (x_n - y_n)^2} \quad (7)\ [6]$$

Where $d(x,y)$ is the euclidean distance, N represents the number of elements in the vectors, n is the current element, and x and y are the given vectors.

The training of the codebooks takes place over a number of epochs. The more epochs the algorithm runs the greater the accuracy at the cost of an increase in computational complexity.

The codebooks generated during the training phase are used to determine the best match for the unknown sample.

3. Materials and Methods

Deep Sample focuses on using an artificial neural network to determine the genre of an audio sample based on the results of a preprocessing step by an audio segmentation algorithm. As such, the implementation and methods span the base segmentation algorithms as well as Deep Sample itself.

3.1 Fast Fourier Transform

The Fast Fourier Transform (FFT) forms the base step for several of the other segmentation algorithms, and therefore was excluded from the analysis by Deep Sample. This was a decision that was made to preserve the independence of the audio segmentation algorithms, thus eliminating any problems with coupling code within the codebase.

The FFT is implemented using a direct FFT model. This model is in place rather than recursive, leading to better performance than a recursive implementation. This is accomplished via a decimation process that replaces the recursive step in the FFT.

3.2 Zero Crossing

The Zero Crossing (ZC) algorithm is implemented in a very similar way. *Equation 2* in the background section shows the equation for ZC. The implementation uses a loop based structure bound by the size of the audio channel. The current implementation supports up to two audio channels, but this is easily extended to accommodate larger audio samples.

3.3 Spectrum Flux

Spectrum Flux corresponds to the timbre of an audio signal. This is determined by converting each channel of the audio wave into a vector of complex numbers, and determining the spectral flux of each vector. Spectral Flux is a summation as given in *Equation (3)*. The results of this summation are then stored in a database for use by Deep Sample.

3.4 Real Cepstrum

Real cepstrum is the product of looping the Fourier transform of each element in a vector. Though, before any calculations are performed on the vector, the data is passed through a hamming window. The window is performed by passing each raw element of the raw vector into the hamming window function as described in *Equation 6*. From there, the function divides the element by a window size. The window size is an arbitrary constant given by how large the sample is. After multiplying by pi, its cosine is taken and returned back to the original cepstrum function. The logarithm of the result is then taken. Finally the vector is then passed through an inverse fourier transform. Refer to *Equation 7* for the mathematical representation of the Fourier transform process.

3.5 Spectrum Centroid

Spectral Centroid is implemented in a similar way to Spectral Flux. First, the Fast Fourier transform of the vector is calculated. Then, the algorithm uses a single loop (for each channel) iterated over the Fourier transform to calculate both summations in the formula (shown in *Equation 4* above). The division is performed at the end.

3.6 Deep Sample

Deep Sample uses the Learning Vector Quantization method to analyze the data. This is accomplished by reading the results of an audio segmentation algorithm from a database. The results are then split into folds for learning. These folds are used to generate a set of random codebooks, which are trained over a specific number of epochs. The trained set of codebooks is used in the best matching unit calculation. In order to get the most varied results for analysis, the folds, codebooks, and epochs were chosen as:

FOLDS	CODEBOOKS	EPOCHS
10	2	5
20	4	10
40	8	20

Table 1: Deep Sample Training Schedule

After the codebooks are trained they are used in the best match unit (BMU) calculation against the unknown sample. The BMU is determined by the shortest euclidean distance, given by *Equation (6)*. This BMU is then used to determine the genre from a lookup table containing the genre of known samples, and the new sample is added to the sample database for use in future analysis.

4. Results and Discussion

Several open source libraries were used to generate samples over the course of the project. All samples were obtained from the Internet Archive,

and only creative commons audio was used. The wave was first loaded and a digital interpretation of the raw signal was created.

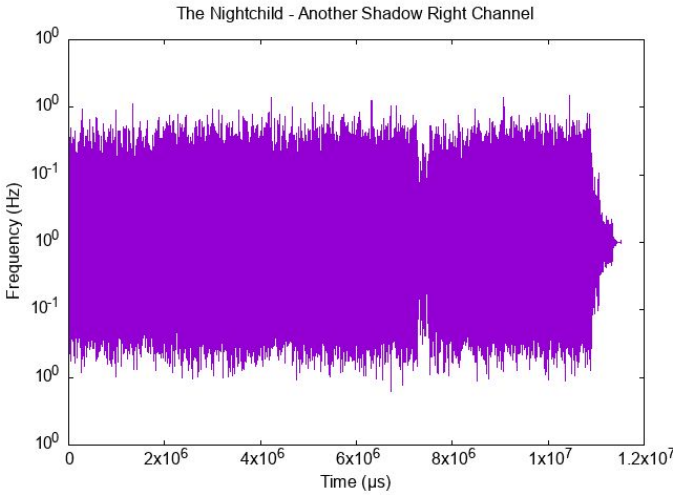

Figure 5: The Nightchild - Another Shadow [7] Left Channel Raw Wave

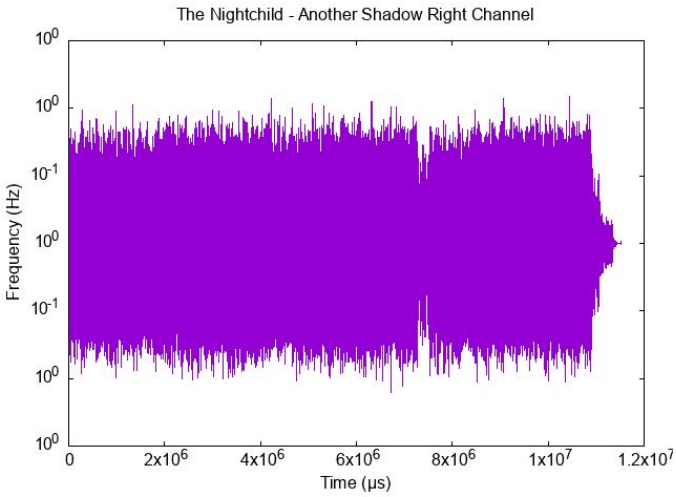

Figure 6: The Nightchild - Another Shadow [7] Right Channel Raw Wave

Figure 5 and Figure 6 show the raw audio wave before it is run through the preprocessing steps. After this wave is loaded, it is run through a Fast Fourier transform yielding the following results.

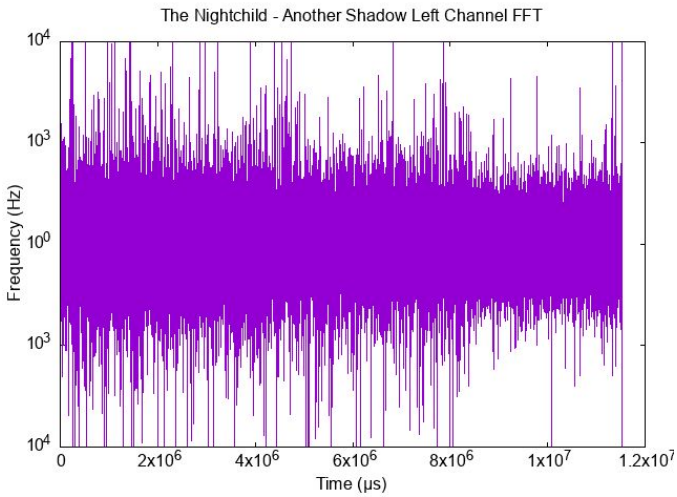

Figure 7: The Nightchild - Another Shadow [7] Left Channel Fast Fourier Transform

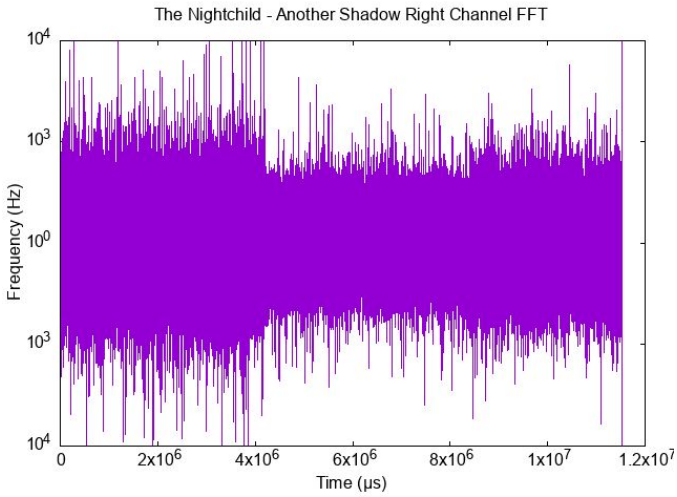

Figure 8 : The Nightchild - Another Shadow [7] Right Channel Fast Fourier Transform

The resulting Fast Fourier transform shown in Figure 6 and 7 was then used as a precursor for the Spectrum Flux, Spectrum Centroid, and Real Cepstrum audio segmentation algorithms.

4.1 Zero Crossing

As discussed previously, the zero crossing algorithm is implemented using a nested loop structure, with each loop being bounded by the size of the current channel. The nature of this boundary leads to a processing complexity of $O(n^2)$. This

results in the zero cross having the poorest performance when looking at it from purely a processing perspective.

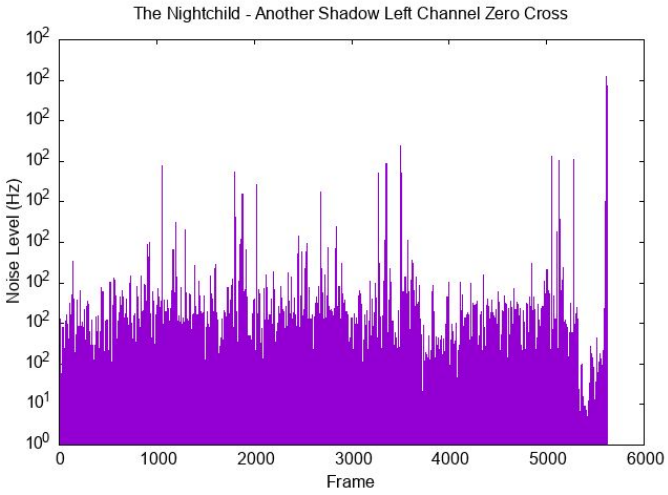

Figure 9: The Nightchild - Another Shadow [7] Left Channel Zero Cross

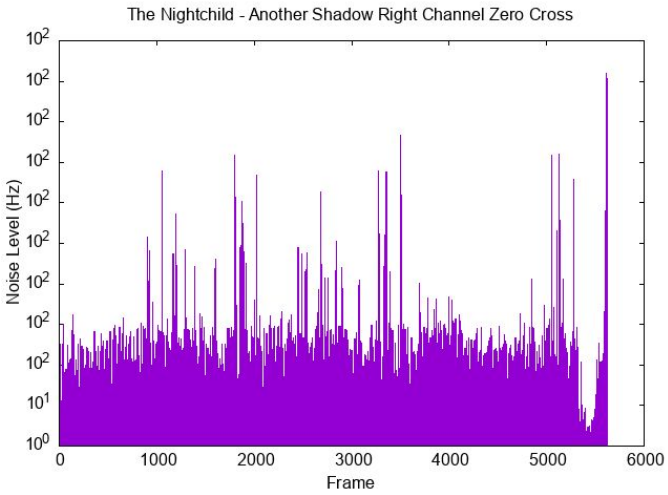

Figure 10: The Nightchild - Another Shadow [7] Right Channel Zero Cross

Figure 9 and Figure 10 show the results of the zero cross algorithm when run on a stereo signal, where the y-axis represents the noise level and the x-axis represents the current time in the audio file. There is a significant level of variation across the signal, meaning that the Zero Cross algorithm is a viable candidate for training and comparison with Deep Sample.

4.2 Spectrum Flux

The spectrum flux algorithm is implemented using a single looping structure bounded by the size of the vector of normals generated by the signal. This results in a linear processing time, O(n), which while not ideal is significantly better than the zero crossing algorithm's $O(n^2)$ performance. Ideally spectrum flux could be improved to run in logarithmic time.

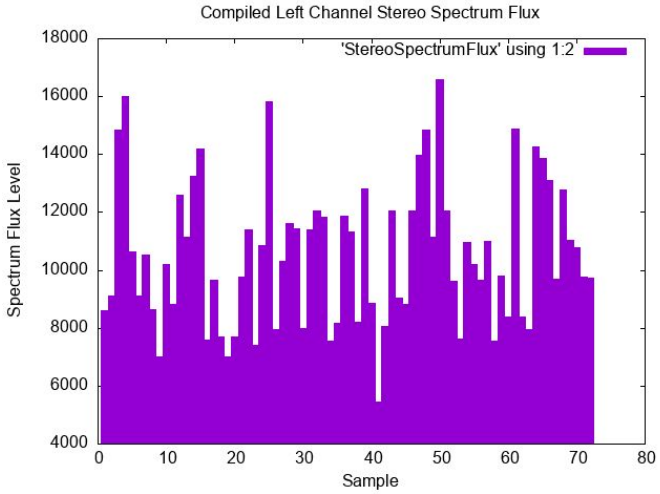

Figure 11: FreeMusic2011 [7] Compiled Left Channel Spectral Fluxes

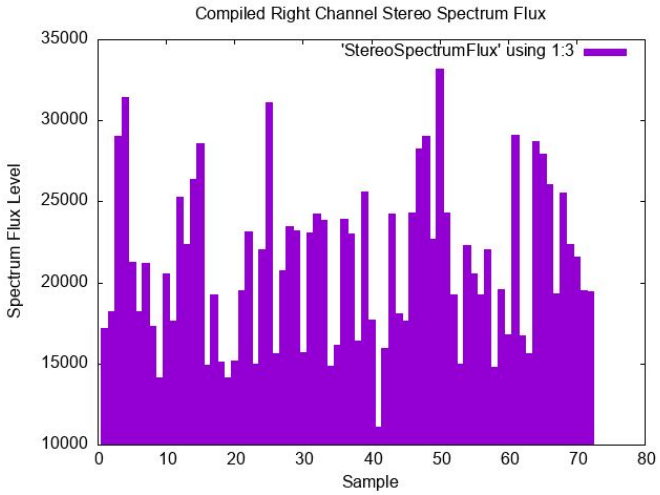

Figure 12: FreeMusic 2011 [7] Compiled Right Channel Spectral Fluxes

Figure 11 and 12 show the results of the spectral flux algorithm run against a database of stereo samples. As can be seen by the graph, the flux of each signal is different enough to allow this measure to be significant when applied to the database. This significance allows for the signal to be acceptable training material for Deep Sample.

4.3 Spectrum Centroid

The spectrum centroid loops once per channel and is bounded by the size of the Fourier transform. Thus, it has a linear processing time of O(n), the same as Spectrum Flux.

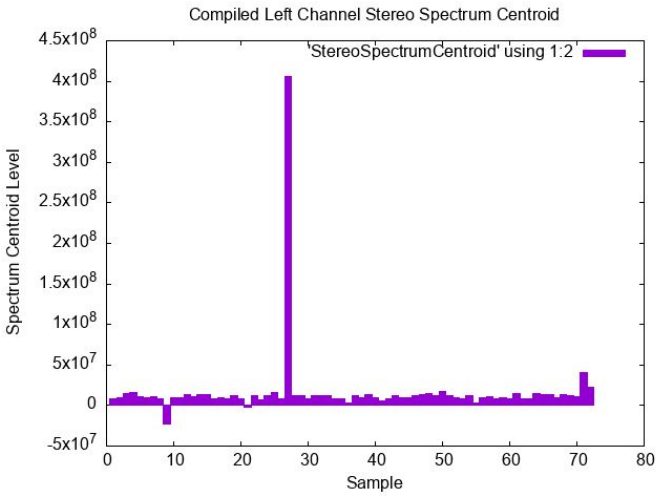

Figure 13: FreeMusic2011 [7] Compiled Left Channel Spectrum Centroids

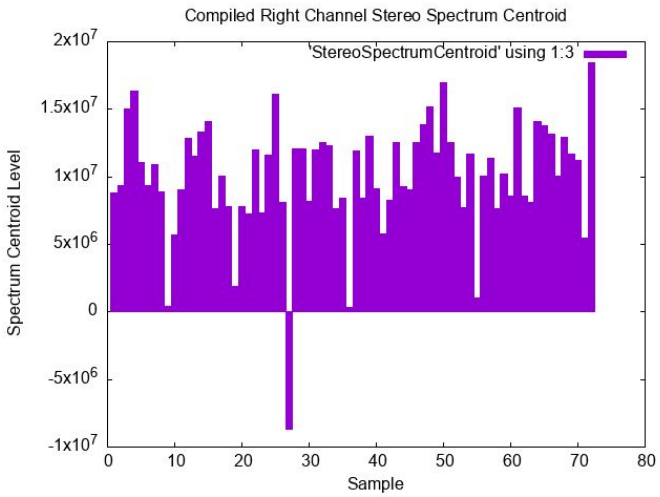

Figure 14: FreeMusic2011 [7] Compiled Right Channel Spectrum Centroids

Figures 13 and 14 show the results of the Spectrum Centroid algorithm when run on a database of stereo samples. The distribution shows a healthy amount of variation which is evident from the graph of the right spectrum centroid, though the variation is eclipsed by a single outlier in the left data set. This variability means that this algorithm, too, is reasonable for training Deep Sample.

4.4 Real Cepstrum

The real cepstrum is calculated by looping through each element of a signal vector and performing basic arithmetic on each element. Performance of the cepstrum algorithm, based on a time complexity, is bounded by the sample size of the vector, therefore having a linear processing time of O(n).

The precursor of the real cepstrum, the hamming window, also performs basic arithmetic and trigonometry to taper off the each end of the sample, in respect to its sample window. Therefore, the processing time complexity, bounded by the sample size, is O(n).

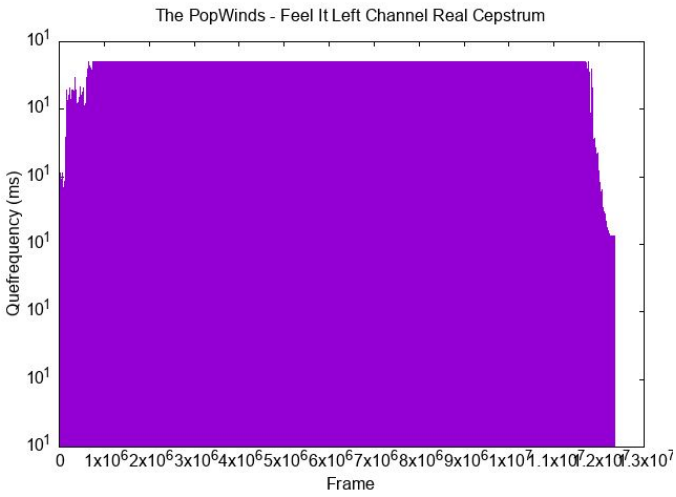

Figure 15: The Pop Winds - Feel It [7] Left Channel Real Cepstrum

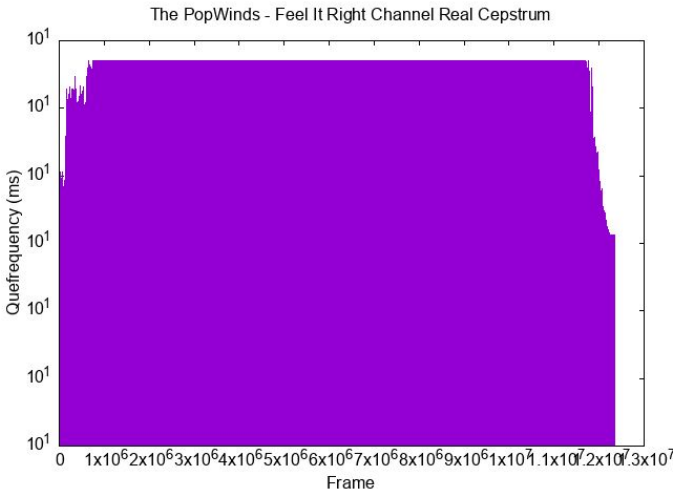

Figure 16: The Pop Winds - Feel It [7] Right Channel Real Cepstrum

The result of performing the real cepstrum on a raw signal vector is shown in Figure 15 and Figure 16. The figures show the graph of each channel, with the y-axis representing the magnitude and the x-axis representing the time domain. The hamming window can be seen as the graph tapers off into either ends.

4.5 Deep Sample

The Deep Sample neural network implements learning vector quantization, which does not use back propagation. As such, the algorithm is bounded by the size of the dataset, making it linear in nature, or O(n).

As discussed in the methodology section, Deep Sample was tested using the training schedule in *Table 1*.

4.5.1 Training Schedule 1

The first training schedule used 10 folds, two codebooks and five epochs to generate the following match for the unknown data sample *Fall Walk Run - You've Got a Way [7]*, hereby referred to as **α**.

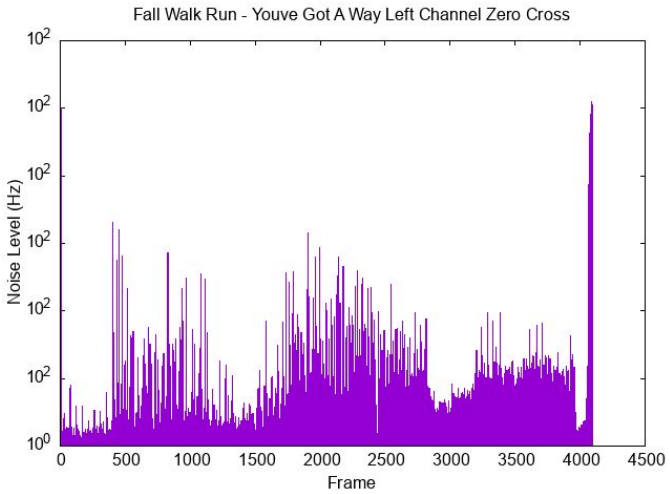

Figure 17: Fall Walk Run - You've Got A Way [7] Left Channel Zero Cross

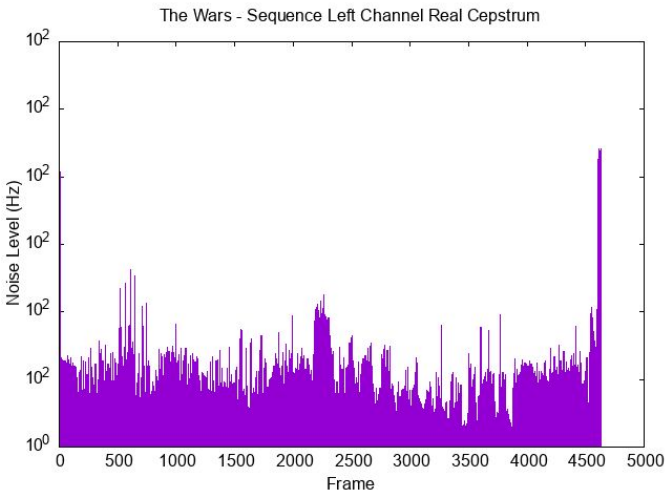

Figure 18: The Wars -Sequence [7]
Left Channel Zero Cross

Figure 17 and Figure 18 show the left channel zero cross of **α**, as well as the left channel zero cross of the proposed match, *The Wars - Sequence [7]***,** hereafter referred to as **β**. Visual inspection of the two shows that their distribution is similar, with both having a similar level of noise across the signal.

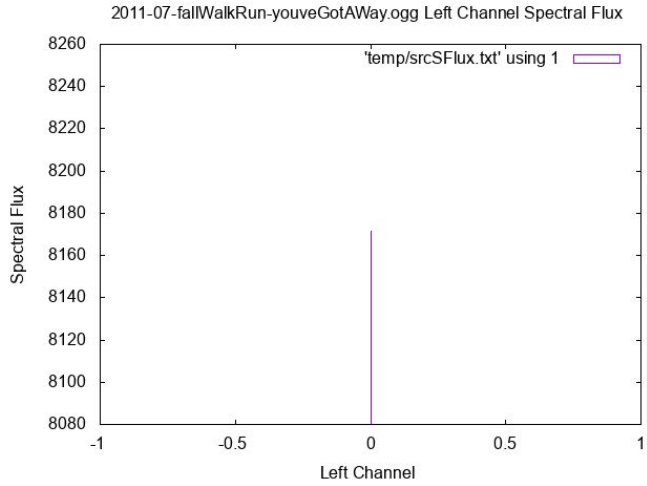

Figure 19: Fall Walk Run - You've Got a Way [7]
Left Channel Spectrum Flux

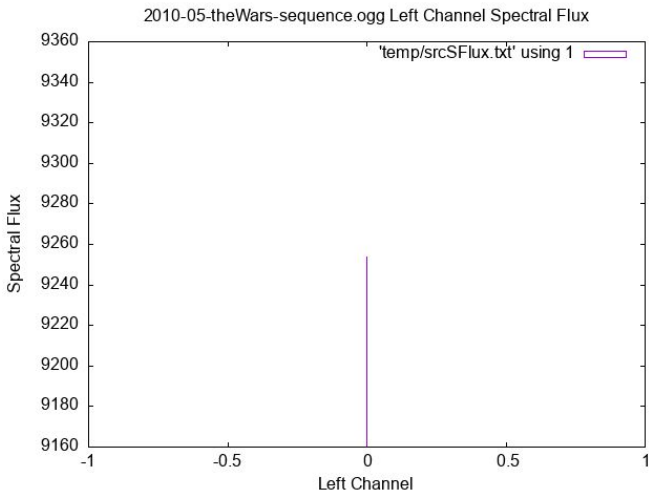

Figure 20: The Wars- Sequence [7]
Left Channel Spectrum Flux

Figure 19 and Figure 20 show the left channel spectrum flux for **α** and **β**. The **α** sample has a spectral flux of 8170 and the **β** sample has a spectral flux of 9250. These can be expressed as 8.17×10^3 and 9.25×10^3, both of which have the same order of magnitude. By having the same order of magnitude it can be concluded that the match given by Deep Sample is a valid match based on the spectrum flux.

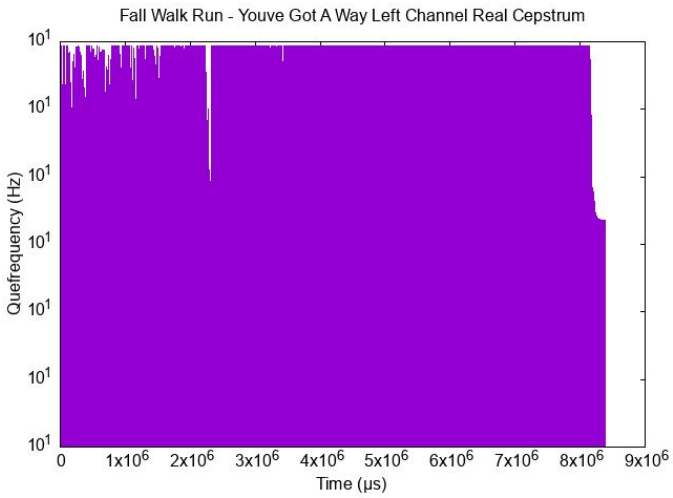

Figure 21: Fall Walk Run - You've Got a Way [7]
Left Channel Real Cepstrum

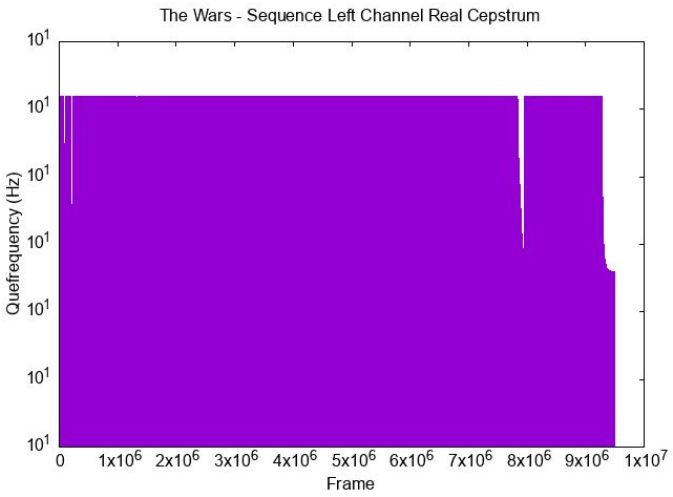

Figure 22: The Wars - Sequence [7] Left Channel Real Cepstrum

Figure 21 and Figure 22 show the left channel real cepstrum for **α** and **β.** The general shape of the graphs are very similar, both fall off at a similar rate. The level of the two cepstrums is fairly equal across both signals, and both of them appear to rise and fall at the same rate. This is enough to conclude through a visual comparison that the match given by Deep Sample using the real cepstrum is valid.

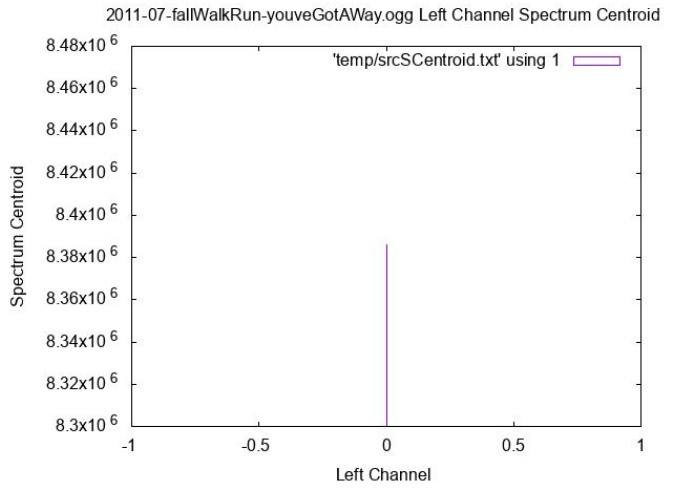

Figure 23: Fall Walk Run - You've Got a Way [7] Left Channel Spectrum Centroid

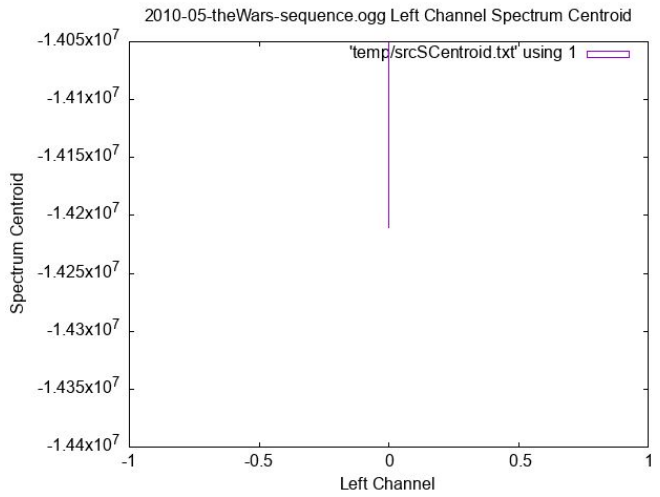

Figure 24: The Wars - Sequence [7] Left Channel Spectrum Centroid

Figure 23 and 24 show the left channel spectrum centroid for **α** and **β**. Due to an outlier being present in the signal, the match by Deep Sample is invalid when using the spectrum centroid for this sample.

According to these results Deep Sample is stating that **α** is similar enough to **β** to be considered the same genre. Based on both visual comparison and examining the difference in order of magnitude of the results, the match is valid using the zero cross, spectrum flux and real cepstrum segmentation algorithms. The results for spectrum centroid have to be discarded as an outlier, meaning that if the prediction was based solely on spectrum centroid it would be invalid. However because the majority of the audio segmentation algorithms returned valid data, this match is still a valid match.

4.5.2 Training Schedule 2

The second training schedule used 20 folds, four codebooks and 10 epochs to generate the following match for the unknown data sample *Studbakers BlackSmith Shop - Elimination of Danger [7]*, hereby referred to as **α**.

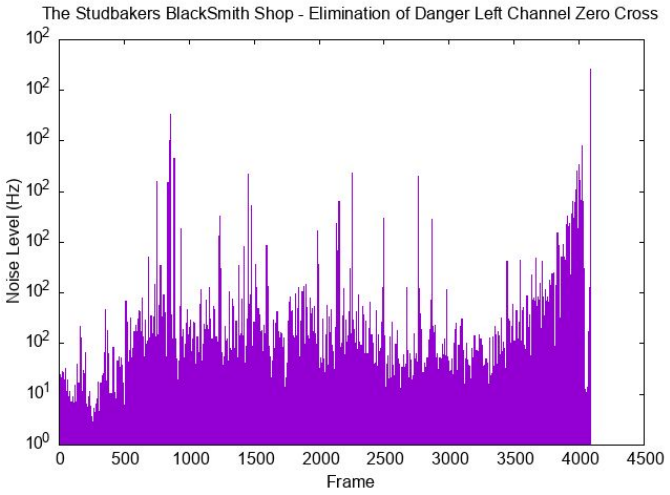

Figure 25: Studbakers BlackSmith Shop - Elimination of Danger [7] Left Channel Zero Cross

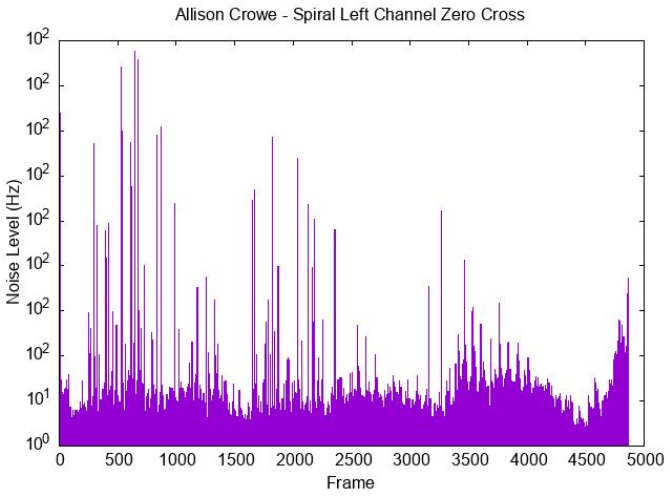

Figure 26: Allison Crowe - Spiral [7] Left Channel Zero Cross

Figure 25 and Figure 26 show the left channel zero cross of **α** along with that of the proposed match *Allison Crowe - Spiral* [7], hereafter referred to as **β**. When compared visually, the two figures show significantly less variation in their peaks than was shown in the comparison between Figure 17 and Figure 18. This is an indication that the increase in folds, training sets and training epochs is having a positive effect.

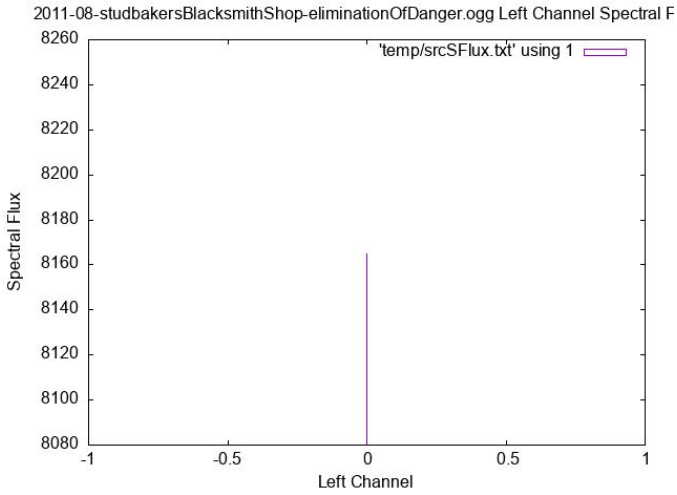

Figure 27: Studbakers BlackSmith Shop - Elimination of Danger [7] Left Channel Spectral Flux

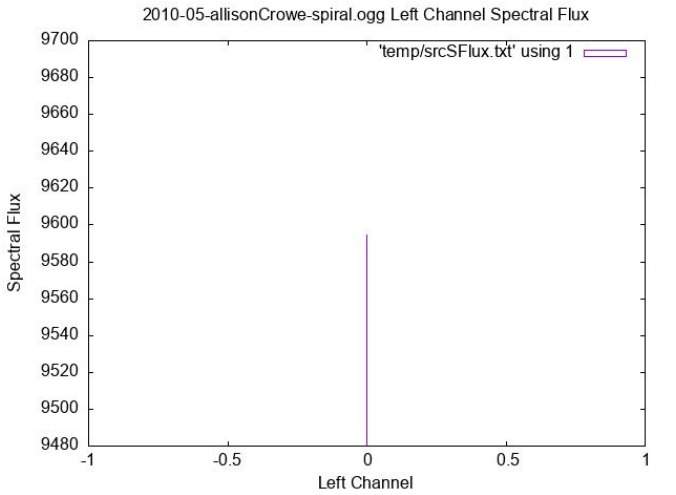

Figure 28: Allison Crowe - Spiral [7] Left Channel Spectral Flux

Figure 27 and Figure 28 show the left channel spectral flux of **α** and **β**. The spectral flux of **α** has a value of 8160, and the value of the spectral flux of **β** is 8590. These can be expressed as 8.16×10^3 and 8.59×10^3. They are of the same order of magnitude as those in the first training schedule, however they are closer together within the set. This shows that the comparison using spectral flux has improved and that the result using this training schedule on the spectral flux algorithm is valid.

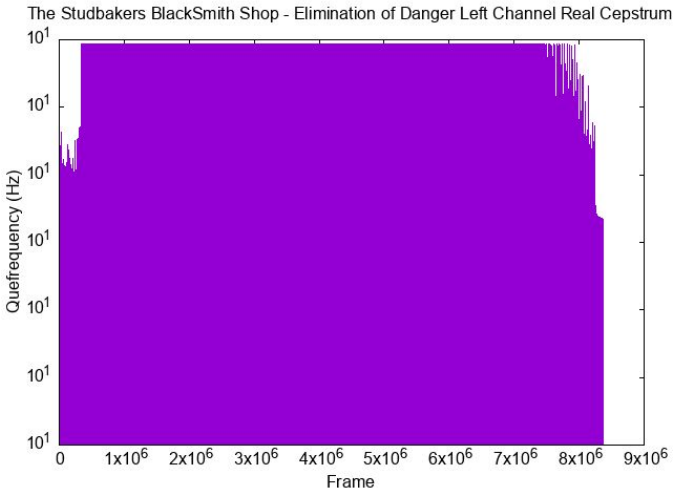

Figure 29: Studbakers BlackSmith Shop - Elimination of Danger [7]
Left Channel Real Cepstrum

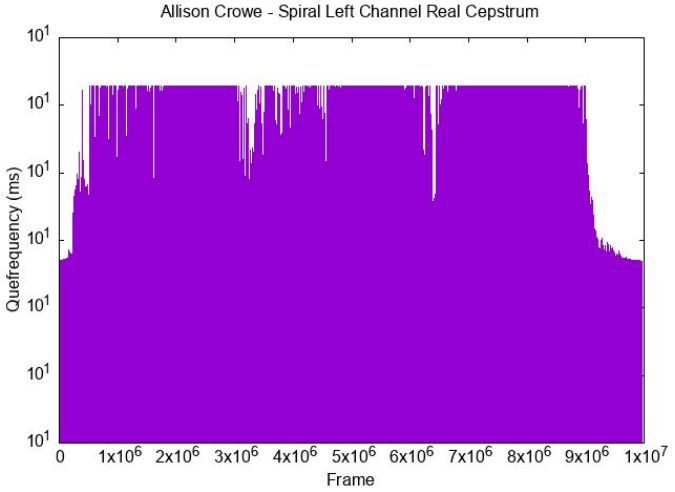

Figure 30: Allison Crowe - Spiral [7]
Left Channel Real Cepstrum

Figure 29 and 30 show the left channel real cepstrum for **α** and **β**. While the general shape of the graphs are similar, the **α** signal appears to fall off a few orders of magnitude before **β**, calling into question the validity of the match when based solely on the real cepstrum.

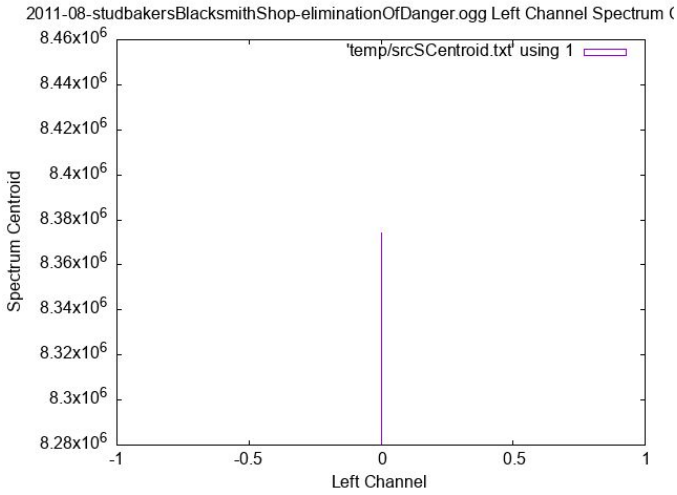

Figure 31: Studbakers BlackSmith Shop - Elimination of Danger [7]
Left Channel Spectrum Centroid

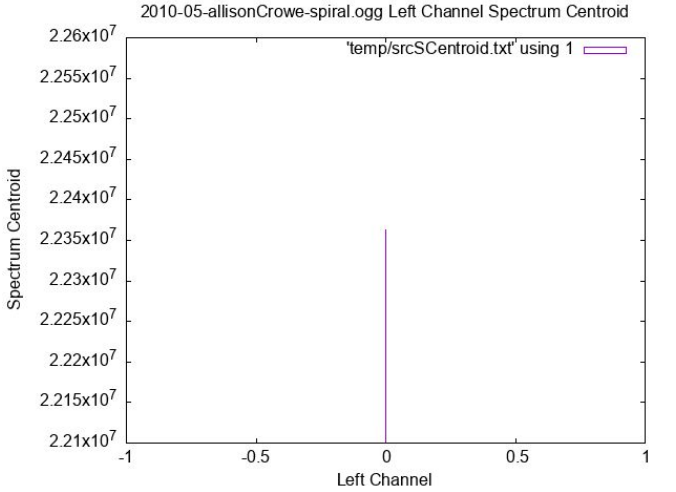

Figure 32: Allison Crowe - Spiral [7]
Left Channel Spectrum Centroid

Figure 31 and Figure 32 show the left channel spectrum centroid of **α** and **β**. The spectrum centroid of **α** is 8.38 x 10^6 and that of **β** is 2.235 x 10^7. These are off by a whole order of magnitude, making the validity of this match less clear when based solely on the spectrum centroid algorithm.

According to these results Deep Sample is stating that **α** is similar enough to **β** to be included in the same genre. These results can be seen by visual comparison of Figure 24 through Figure 31.

The charts can be inspected in a similar manner to those in **Section 4.5.1** to determine that the two samples are a visually close match. Unlike the first test run, the spectrum centroid no longer produces an outlier in the data set. There is also less variation between the two samples, lending credence to the hypothesis that the association between samples increases in accuracy as the number of folds, or neurons, is increased.

In addition to the increase in folds, this training schedule expands the number of available codebooks and the amount of time the network uses to train in the form of a greater number of epochs. The expanded codebook set allows the network to be more discerning when making a selection, and the longer training cycle allows for the error rate in the best match calculation to decrease before a real match is located. All of this translates into better overall performance, in the form of more accurate matches, with the second training schedule.

4.5.3 Training Schedule 3

The third training schedule used 40 folds, eight codebooks and 20 epochs to generate the following match for the unknown data sample *Studbakers BlackSmith Shop - Elimination of Danger [7]*, hereby referred to as **α**.

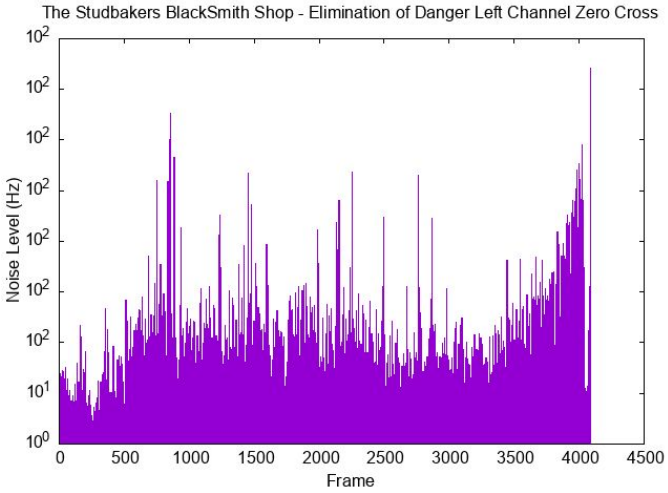

Figure 33: Studbakers BlackSmith Shop - Elimination of Danger [7] Left Channel Zero Cross

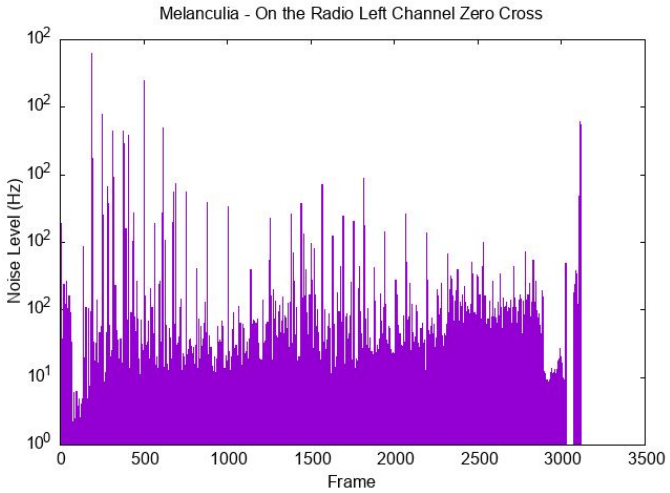

Figure 34: Melenculia - On the Radio [7] Left Channel Zero Cross

Figure 33 and Figure 34 show the left channel zero cross of **α** alongside that of the proposed match *Melenculia - On the Radio* [7], hereafter referred to as **β**. Visual comparison of the graphs shows a similar grouping to that of the second training schedule. This indicates that while there is no visually apparent improvement, the results are still more accurate than those obtained with the first training schedule when using the zero cross algorithm and conclude that the match is valid. This indicates that the increase in folds, codebooks and training epochs is still having a positive effect on the associative capabilities of the neural network.

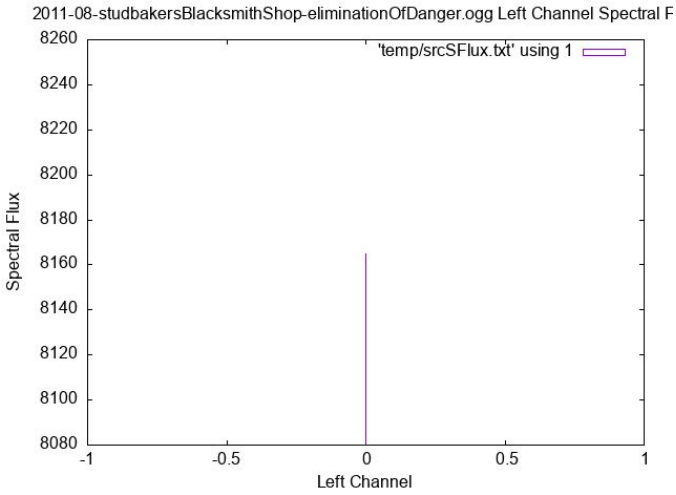

Figure 35: Studbakers BlackSmith Shop - Elimination of Danger [7]
Left Channel Spectral Flux

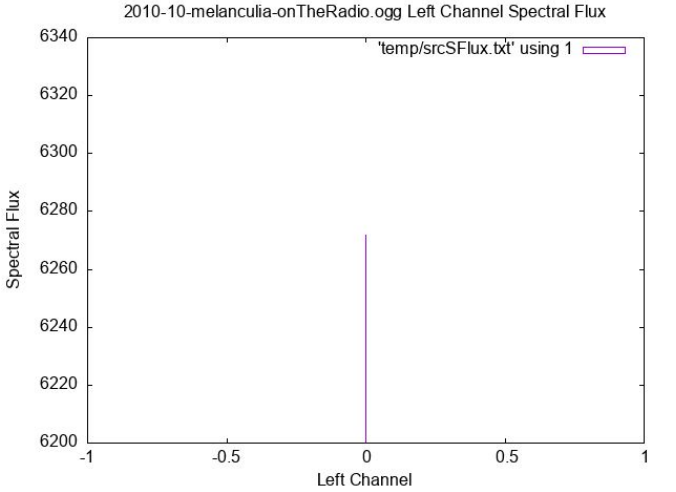

Figure 36: Melenculia - On the Radio [7]
Left Channel Spectral Flux

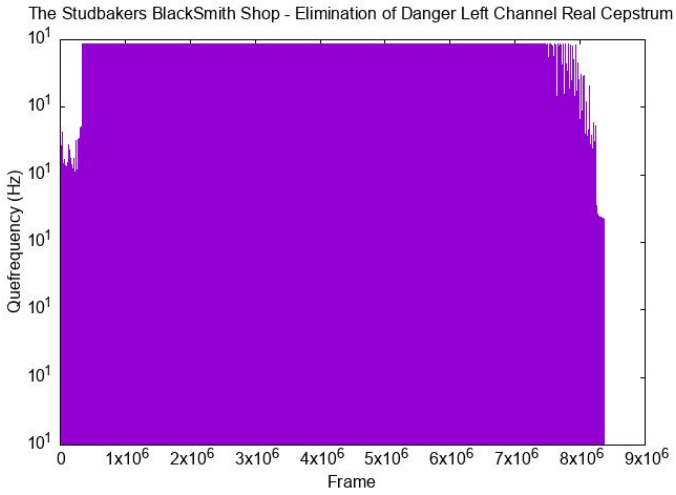

Figure 37: Studbakers BlackSmith Shop - Elimination of Danger [7]
Left Channel Real Cepstrum

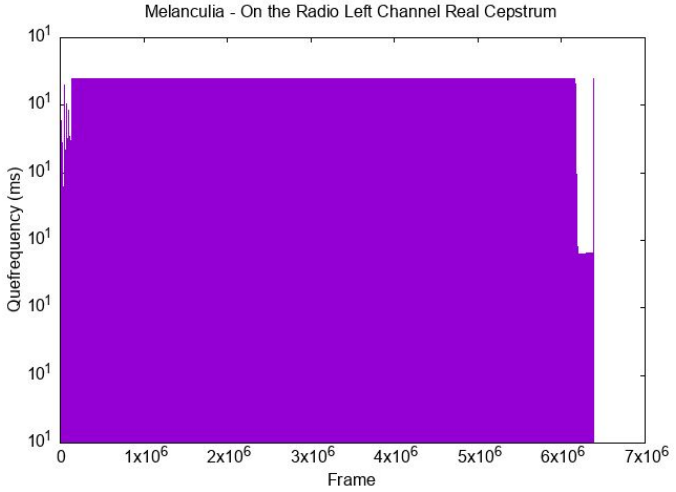

Figure 38: Melenculia - On the Radio [7]
Left Channel Real Cepstrum

Figure 35 and Figure 36 show the left channel spectral flux for **α** and **β**. The spectral flux of **α** is shown to be 8.17 x 10^3 and that of **β** is 6.27 x 10^3. These results are within the same order of magnitude, but are further apart than those of the second training schedule, suggesting that the third training schedule led to a decrease in associative ability when relying on the spectral flux. Despite this decrease in accuracy, the results are still within error tolerance so this is a valid match.

Figure 37 and 38 show the left channel real cepstrum of **α** and **β**. The general shape of the graphs is similar, and there is even less variation between them than was shown in the second training schedule. Both drop off to similar values, lending credence to the validity of this match. The increase in accuracy shows that increasing the folds, codebooks, and training epochs is still having a positive effect on association via the real cepstrum algorithm.

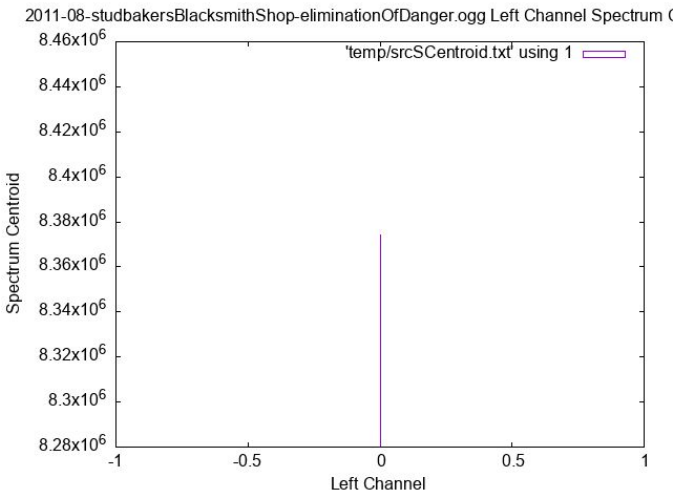

Figure 39: Studbakers BlackSmith Shop - Elimination of Danger [7]
Left Channel Spectrum Centroid

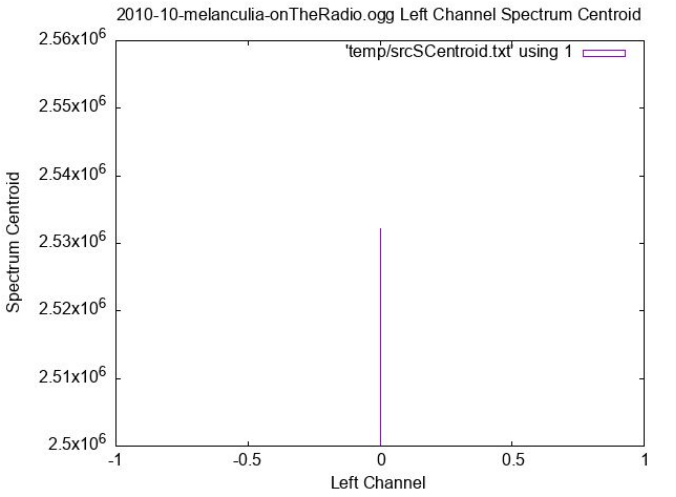

Figure 40: Melenculia - On the Radio [7]
Left Channel Spectrum Centroid

Figure 39 and Figure 40 show the left channel spectrum centroid of **α** and **β**. The spectrum centroid of **α** is 8.38 x 10^6 and that of **β** is 2.53 x 10^6. These are of the same order of magnitude, thus the match is valid. This shows that increasing the number of folds, codebooks, and epochs has a marked positive effect on associations via the spectrum centroid.

According to these results, Deep Sample has determined that **α** and **β** are similar enough to be considered part of the same genre. The charts once again show a decrease in variation between samples, thus indicating that as the number of folds, codebooks and training epochs increases, the ability of Deep Sample to form accurate associations also increases.

5. Conclusion

Listening to each match generated by Deep Sample provided confirmation that the pairs of musical pieces were similar enough in nature to declare the matches valid. As the number of folds, codebooks for training, and training epochs increased, Deep Sample showed a marked improvement in performance. The differences between unknown samples and the known database matches showed a decreasing trend. However, without a larger dataset it is impossible to speak to the overall accuracy of Deep Sample. Increasing the size of the network as well as the amount of training time would also help produce more meaningful data.

One limitation on DeepSample was the inability to transfer data efficiently. Due to the COVID-19 pandemic, large data transfers were impractical through cloud networking, such as Google Drive. Statistical analysis using large datasets were impeded due to inefficiencies. For future works, statistical hypothesis testing is needed on a larger data set to have more conclusive results.

The implementation of the audio segmentation algorithms could be improved through the use of a language designed to be used for signal processing, such as Matlab. The current C++ implementations of the algorithms were a good exercise in learning how to perform audio preprocessing, however they formed a bottleneck in the data analysis process that the project could do without.

Future work should be focused on improving all aspects of Deep Sample, starting with the audio segmentation preprocessing steps. The ability to learn from the dataset also should be added in future iterations of the network.

References

[1] Zahid, S., Hussain F., Rahid M., Yousaf M.H., & Habib H.A. (2015). “Optimized Audio Classification and Segmentation Algorithm by Using Ensemble Methods”. *Mathematical Problems in Engineering*, 2015, 1-11. doi: 10.1155/2015/209814

[2] Steeb, W.H., Steeb W.H. (2005). “Mathematical Tools in Signal Processing with C and Java Simulations”. Hackensack, NJ: World Scientific. doi: 3-8-2020

[3] Giannakapoulos, T., Pikrakis, A. (2014). “Spectral Flux”. Introduction to Audio Analysis. https://www.sciencedirect.com/topics/engineering/spectral-flux

[4] Chu, W.T. (2014). “Musical Genre Classification”. Multimedia Content Analysis. https://www.cs.ccu.edu.tw/~wtchu/courses/2014f_MCA/Lectures/Lecture%209%20Audio%20and%20Music%20Analysis%202.pdf

[5] Tzanetakis, G., Cook, P. (2002). “Musical genre classification of audio signals”. IEEE Trans. On Speech and Audio Processing, vol 10, no. 5, pp. 293-302

[6] Brownlee, J. (2016 Nov 14). How to Implement Learning Vector Quantization (LVQ) From Scratch with Python. machinelearningmastery. https://machinelearningmastery.com/implement-learning-vector-quantization-scratch-python/

[7] Multiple. (2009 Apr 26). Free Music Charts. https://archive.org/details/free-music-charts?tab=collection