Musical Transcription of Drum Patterns Using Main Audio Features as Feature Vector in KNN

Dadios, Charles Jayson L.

BIOGRAPHICAL SKETCH

The author is Charles Jayson L. Dadios. He was born on April 1, 1997, and grew up in Los Baños, Laguna. He graduated from Lopez Elementary School and Los Baños National High School. He would like to pursue his career as a BS Computer Science graduate, developing computer applications which may help the people in their practical struggles.

ACKNOWLEDGEMENT

I would like to thank my parents for supporting me morally and financially. I would like to thank my adviser for his guidance. I would like to thank my best friend and most beloved Joy. If not for her love and grace, I would not have the courage to finish. I thank the Lord, most of all, for all His favor.

ABSTRACT

Abstract—Unlike in chordal instruments, there is limited printed musical information on how to play the drum part of popular songs. This research aims to produce a musical transcription of a drum audio recording in MIDI andPDF format. Each onset from the input was detected andextracted of their main audio features, which were fed into a KNN classifier. The classifier data set and test set came from the audio features of downloaded drum samples that were combined to produce other physiologically available drum sound classes. The classifier performed with the highest overall accuracy of 87% when k=1 for the test set evaluation. Whereas, the actual accuracy for transcribing an input of a common drum pattern was 50%

I. INTRODUCTION

Music has the power to motivate, stir emotions, and even revive a person's memory [1], and in a contemporary setting, it is usually accompanied by a percussion instrument which is mainly the drums. The drums are a rhythmic percussion instrument that is composed mainly of seven main instruments, namely bass drum (kick), snare, tom-toms, floor tom, hi-hat, crash, and ride cymbal. In a band setting, it allows listeners to follow the beat of a composition and helps establish its dynamics and overall feel. It is also known as the backbone of the band along with the electric bass.

Automatic Drum Transcription (ADT) in simple terms is the process of converting a drum performance into a record usually as a drum notation printed as a music sheet.

A. Statement of the Problem

Beginner and professional drummers alike encounter difficulty in learning songs by ear and memorization. This method has been common for folk, rock, and pop musicians [2]. An alternative and more accurate method, one that is common for schooled drummers, would read a drum transcription or chart. Although learning and familiarizing oneself to read charts can be an easy skill to pick up, writing charts takes more time and effort. This challenge encourages drummers to practice the former method, but people tend to forget the correct parts and play inconsistently.

Unlike charts with chords and lyrics for chordal instruments,most popular songs do not have available drum transcriptions,e.g., Figure 1. It would also be difficult and time-consuming for beginners to write charts themselves. But, there is a convenient way to electronically transcribe drum music. It is by using a musical instrument digital interface (MIDI)controller, connected to a Digital Audio Workstation (DAW)that is installed on a computer. However convenient, such devices may not always be practical for common drummers.

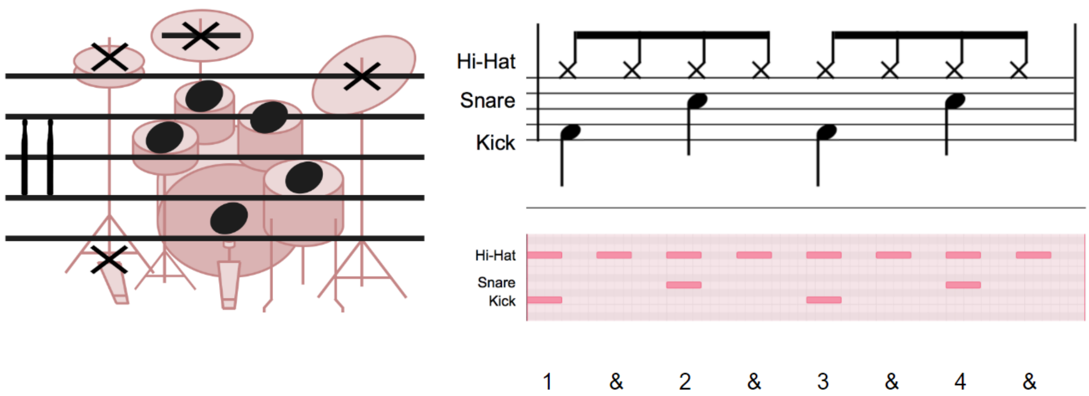

Fig. 1. Sample notation guide, and drum transcription of the song Billy Jean and its midi visualization [18], [19].

B. Significance of Study

Since there are very few songs that have written drum patterns for another player to learn, this study suggests a solution to automate the audio to musical transcription.

C. Objectives

The general objective of this study was to develop an application that can convert a digital drum recording into a musical drum transcription.

Specific:

1) To produce physiologically realistic drum sound combi-nations using downloaded drum instrument audio samples
2) To extract main audio features from the data set
3) To classify the onsets in a given drum recording
4) To produce a PDF and MIDI transcription out of the classified onsets
5) To evaluate the accuracy of the produced transcription

II. SCOPE AND LIMITATION

The audio input tempo and beat can be tracked using the Librosa package for common time signatures, but it's not always successful when given polyrhythmic beats. The detected tempo is sometimes double the original, thus, the correct tempo was made as an optional argument to the program. Beat onsets or hits are also detected but sometimes missed a few when not loud enough.

Another concern for transcription was finding the downbeat the time when to count “1” in a song, as there are no solutions yet to this problem. Thus, for this work, the firstonset in the input recording should be a downbeat, and the time signature should be 4/4, although it is already common in playing drums.

The extracted audio feature dimension was reduced. Somewhere truncated while some were averaged to form a 77-dimensional feature array. The truncation was done to reduce the computational complexity of KNN, and to produce a standard feature length with respect to the start of the note or onset, regardless of the note length.

A manual evaluation to compare the classes of the input vs the output classes was used as there is not yet a solution for the music similarity problem.

II. RELATED LITERATURE

Studies on music transcription have mostly been interested in melodic instruments, but there were still some that focused on ADT, mostly using audio features. [3] The use of audio features was common in automatic music transcription (AMT) [12]. This is also used in ADT but with a different implementation.

FitzGerald identifies ADT as a sound source separation problem. His approach of using simple heuristics was able to identify individual drum parts without any form of rhythmic modeling, where he used the general source separation and redundancy reduction techniques, such as Independent Com-ponent Analysis and Independent Subspace Analysis. Therewere only a few issues of misidentifying a low tom as a kickdrum, and a loud hi-hat being detected as a drum hit [5].

Southall, Stables, and Hockman used Convolutional NeuralNetwork (CNN) in their implementation of ADT. They used spectral features, producing a segmented spectrogram to be the training data for CNN. Their model was trained to identify only the kick drum, snare drum, and hi-hat sounds. It was evaluated using a drum solo, which is a drum mixture (with drum set parts other than kick, snare, and hi-hat), and with a multi-instrument mixture to test their system's ability to adapt from unseen timbres [6].

The study of Miron, Davies, and Gouyon [13] also used fea-ture extraction as feature vectors and the K-Nearest Neighboralgorithm (KNN) for ADT, and they were able to produce analmost 90% accuracy in recognizing the sounded instruments,but the algorithm was intended for live input and only short samples.

In the study of Gillet and Richard, they performed AT from an audio-video file by extracting its audio and visual features. They compared the recognition rate when using only the audio features, only the video features, and joint feature vectors. They found out using the joint feature vectors best among the three [7].

A methodology similar to an undergraduate study [14]was adopted for this study because of their same case of using KNN to classify audio signals using low-level features,but the study focused on heartbeat classification. This study suggested increasing the sample size to improve classification performance.

IV. METHODOLOGY

This study focused on classifying drum onsets concerning time, then producing the transcription. This study was composed of four major processes: data preparation, feature extraction,classification, and transcription. These were implemented using Python 3.

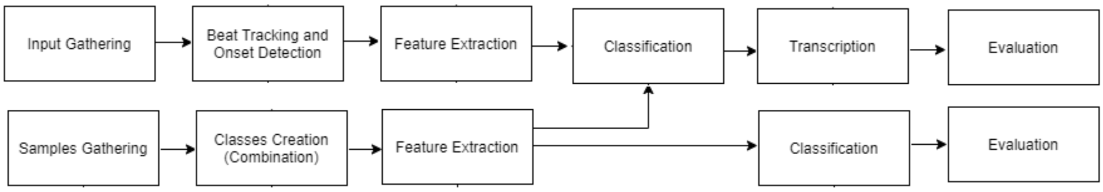

Fig. 2.Methodology

IV. METHODOLOGY

A. DATA GATHERING

There are 7 main kinds of instruments present in a drum kit, namely bass, snare, tom-tom, floor tom, hi-hat, crash, and ride, where three more sounds may be produced, namely open hi-hat, stick, and ride bell, for a total 10 different sounds. Samples of each were downloaded from several websites but mostly from [22], which provided the compilation of initially labeled samples for the single hit classes. All samples were in WAV format of no more than 3 seconds. The single hit class representatives were combined to produce a total of 142 classes, which constituted all the data set. The downloaded single class samples were duplicated—bass (312), bell (323), crash (314), floor (315), hi-hat (314), ride (314), snare (578), stick (312), and tom (583)—to balance the weight of samples for KNN. Meanwhile, other derived classes' samples grew exponentially, having 625 samples each. The script for the combination necessary for producing classes was included in the program.

The single hit classes were combined about the drummer's physiology as seen in Table 1. The complete enumeration of classes is found in Appendix I.

Table 1: Breakdown of the derived classes

Instrument count per beat	Derivation (count)	Subtotal
1	kit pieces (7) + stick, hhopen and bell (3)	10
2	two sticks (36) + one stick and bass (9)	45
3	two sticks and bass (48) + two sticks and foot hihat (20)	68
4	Two sticks, foot hihat and bass (19)	19
total classes		142

B. Feature Extraction

Feature extraction is the process of creating a new set of k-dimensional features from the original data [4]. It is done to reduce the computational complexity of signal processing algorithms [9]. The feature set used for classification were extracted using LibROSA [20], a Python package for music and audio analysis.

The feature set was composed of low-level audio features which were readily computed abstractions that LibROSA provides for processing raw audio inputs. The following features were used:

1. *Spectral bandwidth* describes the density of the signal. It can be computed by calling the function librosa.feature.spectral_bandwidth(y), where y is the audio time series array.
2. *Spectral contrast* describes the difference of the high and low points of the audio signal’s spectrum, computed as librosa.feature.spectral_contrast(y).
3. *Zero crossing rate* describes how often the audio signal crossed the 0

voltage axis, computed as librosa.feature.zero_crossing_rate(y).

4. *Mel-Frequency Cepstral Coefficient* which represents the shape of the spectrum, as to how a human auditory perception would react with very few coefficients, computed as librosa.feature.mfcc(y).
5. *Spectral centroid* which describes the central mass of the input signal, computed by calling the function librosa.feature.spectral_bandwidth(y).
6. *Spectral flatness* is the quantification of a sound from being noise-like versus being tone-like, computed as lbrosa.feature.spectral_flatness(y).
7. *Spectral roll-off* describes the distribution of the power along the frequency spectrum, and is computed as librosa.feature.spectral_rolloff(y).

Librosa returned the features as a Numpy array. The MFCC and spectral contrast were averaged as an array of lengths 20 and 7 respectively, while the remaining features were truncated to be 10 elements long, forming a total of 77-dimensional feature array.

C. Classification

The algorithm used the Scikit-learn package for KNN [21] performed classification over the data set. 10% of the data was partitioned as the validation set. The evaluation for classifying individual samples was done using a built-in abstraction of the package for evaluating KNN accuracy.

KNN decreases in performance as it increases the feature dimension, so the data set size was increased to compensate [16].

The algorithm was utilized to perform the classification for longer audio streams of drum beat patterns. The audio input was segmented by the number of its onsets using the Librosa onset detection abstraction. Onset positions were recorded regarding beat locations using the Librosa beat tracker. Each onset was fed into the classifier. Onset classifications concerning tracked beats were transcribed using a script included in the program. The output was produced as a MIDI format file using the Python MIDIUtil library, and a PDF format using Lilypond [23].

B. Evaluation

The algorithm used for classifying the sounds was evaluated for its accuracy in correctly identifying the sounding classes. The Scikit-learn has a built-in library for evaluating KNN accuracy.

Also, the full transcription algorithm was run and evaluated using two types of input:

1. A briefly composed midi drums played through LMMS (digital audio workstation) and imported as a wav audio file
2. A recorded solo drums performance

A manual comparison was done, through counting the different classes, to compare the actual classes from the input audio, versus those from the output transcription. By this method, a confusion matrix was created by examining the output transcription. The confusion matrix was used to visualize and compute the precision, recall, and accuracy of the algorithm. Precision describes the percentage of relevant results being correctly classified, while recall is the percentage of relevant results. Accuracy, on the other hand, describes how correct are the results. Figure 3 shows the formulas for their computation. [17]

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

Figure 3. Formula for computing precision, recall, and accuracy [17]

V. RESULTS AND DISCUSSIONS

The algorithm was successful in combining basic drums audio samples to produce more physiologically realistic sample classes. Out of the samples, main audio features were extracted for each. Since the combination took about 3 hours and 10 GB of memory to complete in a single process script, the produced samples were excluded from the source. However, the extracted features were included in the source code as comma-separated values (CSV). The CSV was for loading every transcription run, instead of recombining or re-extracting audio and features.

The samples produced were used in KNN to compute for accuracy. 30% of the samples were used for test evaluation. KNN accuracy was compared for three sets of classes, namely *simple*, *common*, and *all*.

The *simple* class set on the table pertains to the sound classes bass, snare, and hi-hat which are the most commonly heard in pop songs. The *common* class set pertains to most parts of the drum kit, except that the least common sound combinations were eliminated. The *all* class pertains to the total physiologically possible drum sound combinations.

The *simple* class set consists of the following classes:

1. bass
2. snare
3. hihat
4. hihat, bass
5. hihat, snare
6. bass, snare

The *common* class set consists of the combinations of the following classes:

1. bass
2. snare
3. tom
4. floor
5. hihat
6. crash
7. ride

minus these less commonly used combinations:

1. bass, tom
2. bass, floor
3. snare, tom
4. snare, crash
5. tom, floor
6. tom, hihat
7. tom, crash
8. floor, hihat

9. floor, crash
10. hihat, crash
11. hitat, ride
12. crash, ride

The *all* class set consists of the all the possible combinations of the following classes:

1. bass
2. snare
3. tom
4. floor
5. hihat
6. crash
7. ride
8. stick
9. hihat-open
10. bell

The full list of classes for \textit{all} class set is found in Appendix I.

The considered classes were adjustable through whitelisting or blacklisting a sound, or removing specific combinations.

The KNN classifier which used the audio features was evaluated with the highest accuracy of 98% when identifying just three classes instead of the full 142, and when k=1. The classifier was also evaluated in varying constraints—different number of cases considered, and different values of k—and getting the average of 5 test runs each. Table 2 shows their accuracy.

Table 2: KNN accuracy in varying k value and number of classes

Class set	k=1	k=3	k=5	k=7	k=9
Simple	98.29%	95.44%	93.15%	91.41%	91.68%
Common	97.03%	93.59%	90.66%	88.26%	86.54%
All	87.41%	75.40%	68.35%	63.91%	60.82%

B. Transcription

Two live and one digital drum tracks were used as input to test the performance of the transcription algorithm. The table 3 and 4 show results of the former which use bass, snare, and hi-hat patterns; while table 5 shows the result of the latter which uses rhythm patterns of bass, snare, hi-hat, open-hi-hat, tom, floor, crash, and ride. The PDF output for tables 3 and 4 are shown in appendix 2.

Here are the sample sizes of the tables.

Table 3: bass 16, snare 8, hihat 16

Table 4: hihat 65, snare 22, bass 17

Table 5: bass 36, snare 20, hihat-closed 26, hihat-open 7, crash 4, ride 20, tom 4, floor 4

Table 3: Confusion matrix of bass, snare and hihat transcriptions (truth data vs classifier results[1])

Class	bass	snare	hihat
bass	8	0	5
snare	0	4	0
hihat	2	4	4

Ovarall accuracy 59.26%

Table 4: Confusion matrix of bass, snare and hihat transcriptions (truth data vs classifier results[1])

Class	bass	snare	hihat
bass	16	1	5
snare	4	18	0
hihat	5	23	51

Ovarall accuracy 69.11%

Table 5: Confusion matrix for commonly used classes (truth data vs classifier results)

Class	bs	sn	hh	hho	tm	fl	cr	rd
bass	22	7	5	1	2	3	2	1
snare	5	15	1	0	0	1	1	0
hihat	0	7	18	1	0	0	0	0
hhopn	0	0	0	7	0	0	0	1
tom	2	0	0	0	0	1	0	1
floor	2	0	0	0	2	0	0	0
crash	0	0	0	1	0	0	3	0
ride	0	1	16	0	0	0	2	2

Ovarall accuracy 50.38%

Compared with the evaluation of the initial KNN test set, the actual performance when using the input recordings was less accurate. Still, the application results were good since most of the precision and recall levels were above 50%.

Two main factors were considered to affect the classifier performance: the sampling environment and the drum audio features themselves. The KNN dataset samples were recorded from a controlled environment, usually to minimize ambient noise. Whereas, the recorded inputs for transcription had more reverberation, thus affecting the succeeding onsets on another class. Also, the input being a continuous stream, the features of the currently evaluated class was affected by the previous class, whose features leaked to the current. This was most evident for recordings with a fast tempo. Also, the segmentation of the input based on individual onsets was not as aligned as the controlled samples. The second factor was that drum set sounds have similar features, so depending on the audio dynamics and how the audio levels were mixed when some audio classes were loud enough, other audio features got overwhelmed by other classes and thus missed.

C. Output

There were two process outputs during the program. One is the produced audio classes by combining individual drum kit sounds. The other is the file output of the transcription. In Appendix III fig. 10 is the average waveform of a bass class. Figure 11 is the average waveform of the bell class. Figure 12 is the average waveform of the classes bass and bell combined. The rest of the figures in Appendix III are the waveforms of the single classes.

Each class was extracted of its features. The visualization of the features of some classes is found in Appendix IV. Unlike the waveform, the trend of the classes represented in their extracted features is barely obvious, but KNN can make sense of it for classification.

A notation of an audio input “bass_snare_hihat_loop” (Appendix II fig. 4) when transcribed by the program produced theAppendix II fig. 5. To aid in reading the output transcription, lyrics for counting were added. At the first mention of the lyric “a" for ”bass_snare_hihat_loop”, the hi-hat hit (marked with”x”) from the count ”One” was not dropped—the “x” was not dropped. This could be due to the features of the previous drum hit/onset leaking to the features of the succeeding drum hit/onset. Because the input audio has reverberation, the cymbal hit did not fade out before another drum hit/onset was made.

Appendix II figures 6 and 7 are the MIDI equivalent of the previously mentioned notations.

Another probable cause of the error is the tempo of the input. Similar to the case of reverberation in the input, when the tempo is fast enough, the features of the previous hit could have leaked to its succeeding hit. This could be the case for “slow_beat” (Appendix II fig. 8), where a hi-hat hit (marked with “x”) was thought to be a hi-hat and a snare hit/class(Appendix II fig. 9 at the first mention of the lyric “n”). The Trailing low frequency of the bass class (Appendix II fig. 9,second “One” lyric) was mistaken as a snare hit.

VI. CONCLUSION

The program successfully combined the drum samples to produce other physiologically valid classes.

Audio features were successfully extracted from every class sample.

Using main audio features in KNN is effective in identifying individual onset types from another, attaining the highest overall accuracy of 98% for k=1 and when considering only between 3 classes, even with the feature vector dimension of 77, in this case.

The algorithm was able to output the same transcriptions but in both

PDF and MIDI format.

The transcription performance was evaluated with an accuracy of 50% for k=1 when considering the commonly used classes, and the highest of 85% when considering just the three classes.

VII. RECOMMENDATION

Just like speech recognition, ADT may improve with the help of other machine learning techniques like the Hidden Markov Model or Artificial Neural Networks.

REFERENCES

[1] E. Mannes, "The Power of Music: Pioneering Discoveries in the New Science of Song"

[2] Ginsborg et al., "Musical Excellence: Strategies and Techniques to Enhance Performance," p. 123, 2004.

[3] C.-W. Wu, "A Review of Automatic Drum Transcription," 2017.

[4] E. Alpaydin, "Introduction to Machine Learning Second Edition," 2010.

[5] D. FitzGerald, "Automatic Drum Transcription and Source Separation," 2004.

[6] C. Southall, R. Stables, and J. Hockman, "Automatic Drum Transcription For Polyphonic Recordings Using Soft Attention Mechanisms And Convolutional Neural Networks," 2017.

[7] O. Gillet, and G. Richard, "Automatic Transcription Of Drum Sequences Using Audiovisual Features," 2005.

[8] M. Sokolova, and G. Lapalme, "A systematic analysis of performance measures for classification tasks," 2009.

[9] J. Breebart, and M. F. McKinney, "Features for Audio Classification," 2004.

[10] C.-F. Liao, "Understanding the CMU Sphinx Speech Recognition System," 2009

[11] M. Miron, M. E.P. Davies, and F. Gouyon, "An Open-source Drum Transcription System For Pure Data And Max MSP," 2013.

[12] McKinney, Martin, and Jeroen Breebaart. "Features for audio and music classification." (2003).

[13] Miron, Marius, Matthew EP Davies, and Fabien Gouyon. "An open-source drum transcription system for pure data and Max MSP." 2013 IEEE

International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013.

[14] C. C. L. Cepe, and M. A. A. Clariño, "Heartbeat Classification using Low-Level MFCC Features with Naive Bayes and KNN," 2018.

[15] Kekre, H. B., and Kavita Patil. "Standard deviation of mean and variance of rows and columns of images for CBIR." International Journal of Computer, Information, and Systems Science, and Engineering 3.1 (2009).

[16] Shetty, Badreesh, and Badreesh Shetty. “Curse of Dimensionality.” Towards Data Science, Towards Data Science, 15 Jan. 2019, towardsdatascience.com/curse-of-dimensionality-2092410f3d27.

[17] Saxena, S. "Precision vs Recall." Towards Data Science. 2018, https://towardsdatascience.com/precision-vs-recall-386cf9f89488.

[18] https://i.redd.it/qo06bda8mjh41.png

[19] Payne, W., and Ruthmann, A., "Tagged Scratch" https://jitp.commons.gc.cuny.edu/tag/scratch/

[20] B. McFee et al., "librosa: Audio and Music Signal Analysis in Python," 2015.

[21] Pedregosa, F., et al, "Scikit-learn: Machine Learning in {P}ython," 2011.

[22] https://freewavesamples.com/

[23] lilypond.org

APPENDIX 1. ENUMERATION OF CLASSES

The following figures would enumerate classes that are appropriate to the physiology of a drummer.

Instrument pieces (7) and available sounds (3)

1. bass
2. snare
3. tom
4. floor
5. hihat
6. crash
7. ride
8. stick
9. hihat-open
10. bell

One stick and bass (9)

1. bass snare
2. bass stick
3. bass tom
4. bass floor
5. bass hihat
6. bass hihat-open
7. bass crash
8. bass ride
9. bass bell

Two sticks (36)

1. snare stick
2. snare tom
3. snare floor
4. snare hihat
5. snare hihat-open
6. snare crash
7. snare ride
8. snare bell
9. stick tom
10. stick floor
11. stick hihat
12. stick hihat-open
13. stick crash
14. stick ride
15. stick bell
16. tom floor
17. tom hihat
18. tom hihat-open
19. tom crash
20. tom ride
21. tom bell
22. floor hihat
23. floor hihat-open
24. floor crash

25. floor ride
26. floor bell
27. hihat crash
28. hihat ride
29. hihat bell
30. hihat-open
31. hihat-open crash
32. hihat-open ride
33. hihat-open bell
34. crash ride
35. crash bell
36. ride bell

Two sticks and bass (48)

1) bass snare tom
2) bass snare floor
3) bass snare hihat
4) bass snare hihat-open
5) bass snare crash
6) bass snare ride
7) bass snare bell
8) bass stick hihat
9) bass stick hihat-open
10) bass stick tom
11) bass stick floor
12) bass stick hihat

13) bass stick hihat-open
14) bass stick crash
15) bass stick ride
16) bass stick bell
17) bass tom hihat
18) bass tom hihat-open
19) bass tom floor
20) bass tom hihat
21) bass tom hihat-open
22) bass tom crash
23) bass tom ride
24) bass tom bell
25) bass tom bell hihat
26) bass floor hihat
27) bass floor hihat-open
28) bass floor hihat
29) bass floor hihat-open
30) bass floor crash
31) bass floor ride
32) bass floor bell
33) bass hihat hihat-open
34) bass hihat crash
35) bass hihat ride
36) bass hihat bell
37) bass hihat-open crash

38) bass hihat-open ride

39) bass hihat-open bell

40) bass crash hihat

41) bass crash hihat-open

42) bass crash ride

43) bass crash bell

44) bass ride hihat

45) bass ride hihat-open

46) bass ride bell

47) bass bell hihat

48) bass bell hihat-open

Two sticks and foot hihat (20)

1) snare stick hihat

2) snare tom hihat

3) snare floor hihat

4) snare crash hihat

5) snare ride hihat

6) snare bell hihat

7) stick tom hihat

8) stick floor hihat

9) stick crash hihat

10) stick ride hihat

11) stick bell hihat

12) tom floor hihat

13) tom crash hihat

14) tom ride hihat

15) tom bell hihat

16) floor crash hihat

17) floor ride hihat

18) floor bell hihat

19) crash ride hihat

20) crash bell hihat

21) ride bell hihat

Two sticks, foot hi-hat and bass (19)

1) bass snare tom hihat

2) bass snare floor hihat

3) bass snare crash hihat

4) bass snare ride hihat

5) bass snare bell hihat

6) bass stick tom hihat

7) bass stick floor hihat

8) bass stick crash hihat

9) bass stick ride hihat

10) bass stick bell hihat

11) bass tom floor hihat

12) bass tom crash hihat

13) bass tom ride hihat

14) bass floor crash hihat

15) bass floor ride hihat

16) bass floor bell hihat

17) bass crash ride hihat

18) bass crash bell hihat

19) bass ride bell hihat

APPENDIX II. TRANSCRIPTION

Shows the comparison of the correct transcription of input vs the actual output transcription.

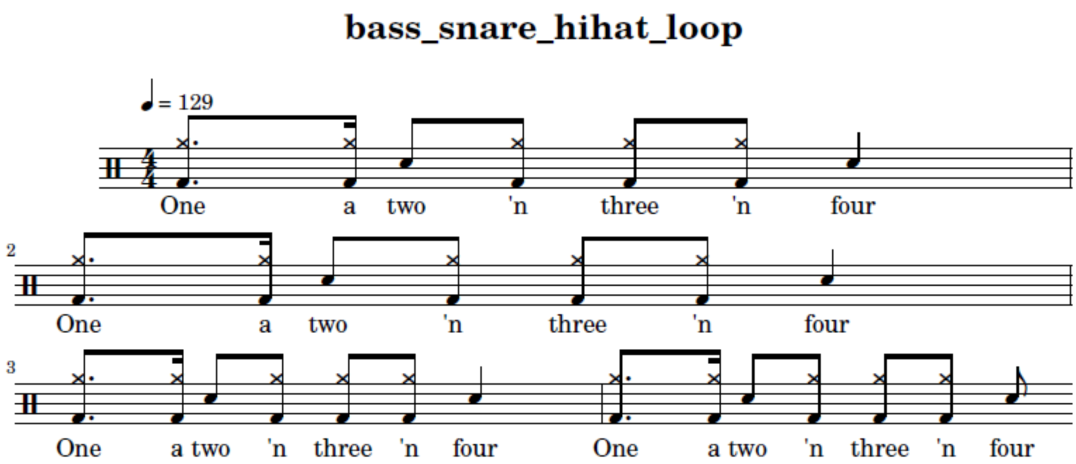

Fig. 4. Correct transcription of table 3 bass_snare_hihatloop.wav

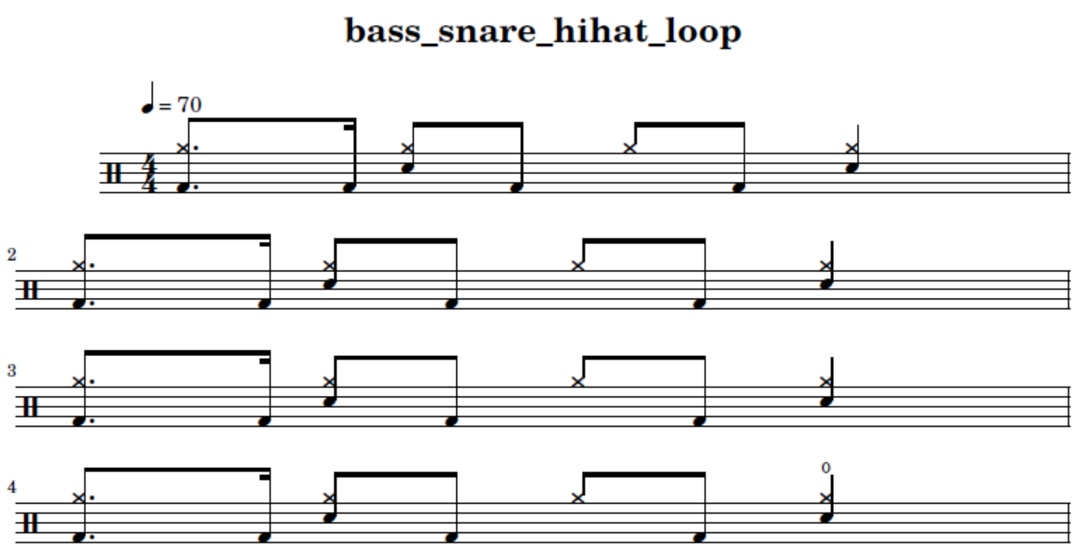

Fig. 5. Actual transcription of table 3 bass_snare_hihat_loop.wav

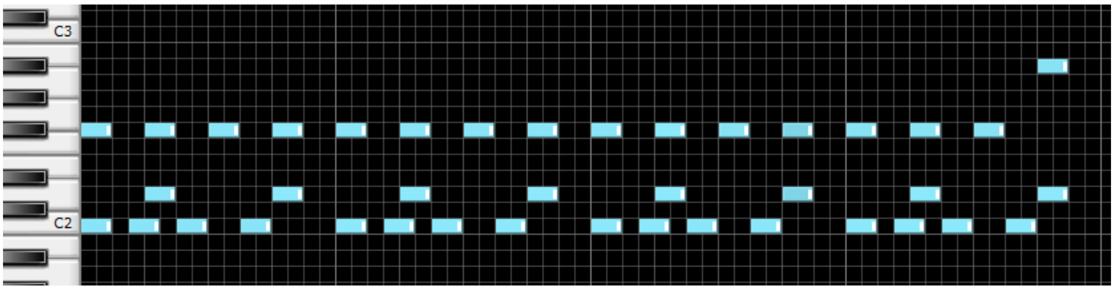

Fig. 6. Correct midi visualization of figure 4 through LMMS

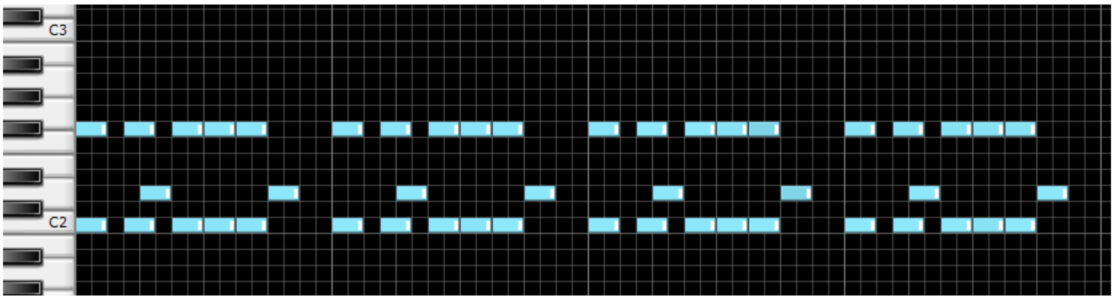

Fig. 7. Actual midi visualization of figure 5 through LMMS

Fig. 8. Correct transcription of table 3 slow_beat.mp3

Fig. 9. Actual transcription of table 3 slow_beat.mp3

APPENDIX III. SAMPLED DRUMS AVERAGE

The averaged visualization of the class sound wave

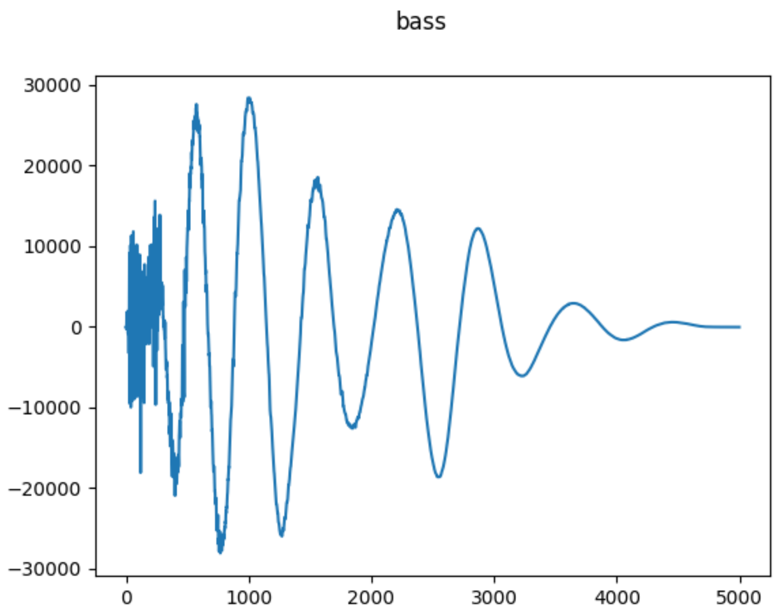

Figure 10. The average waveform of class bass

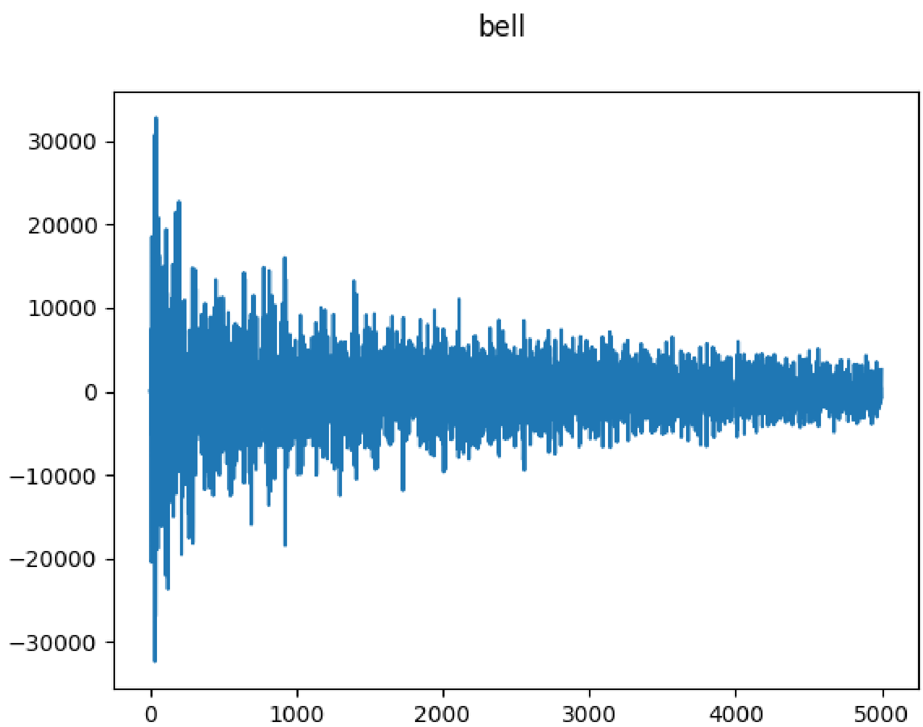

Figure 11. The average waveform of class bell

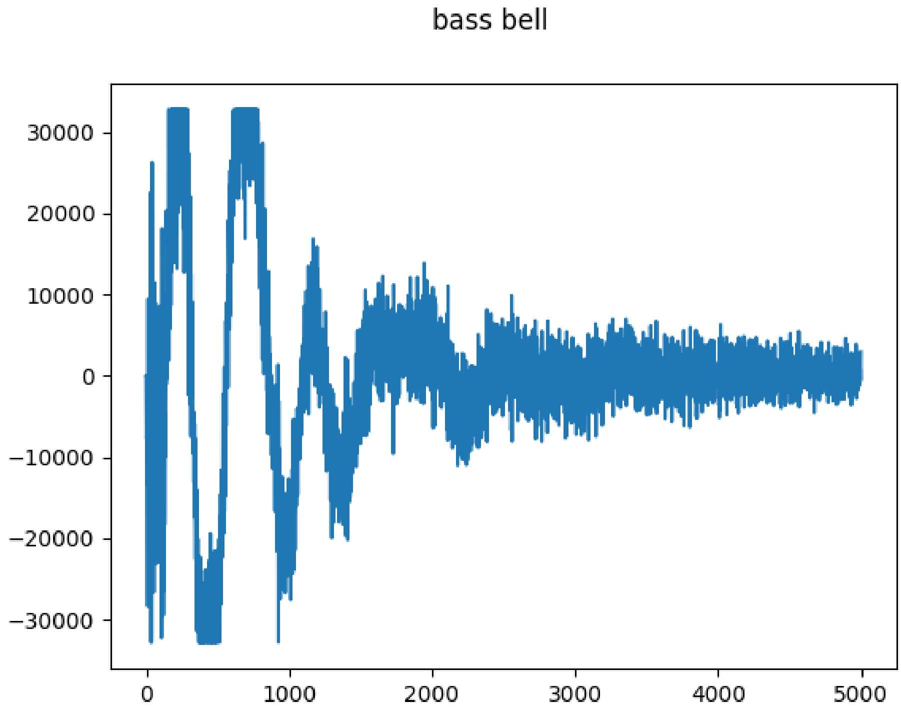

Figure 12. The average waveform of class bass bell

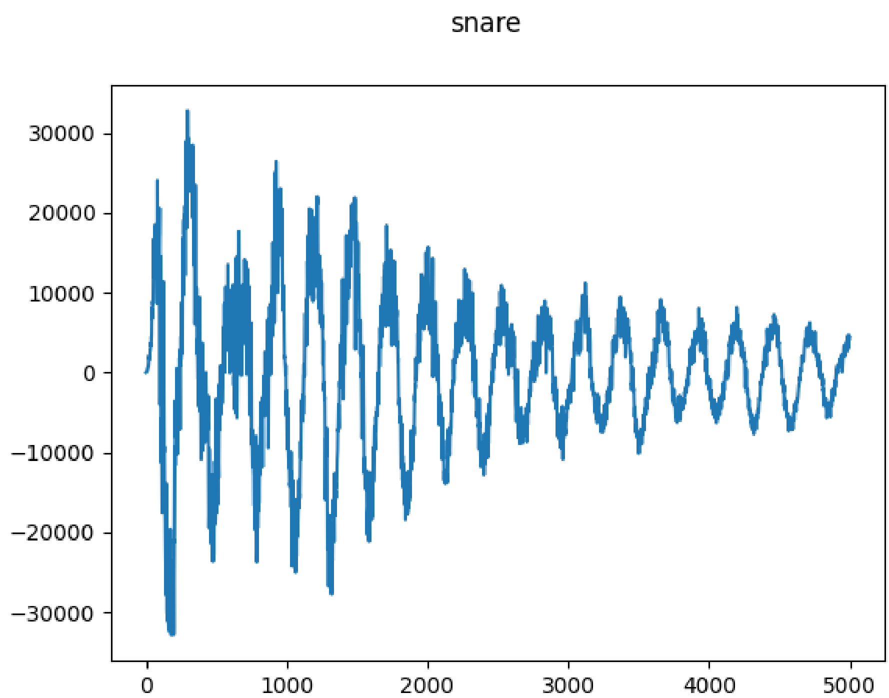

Figure 13. The average waveform of class snare

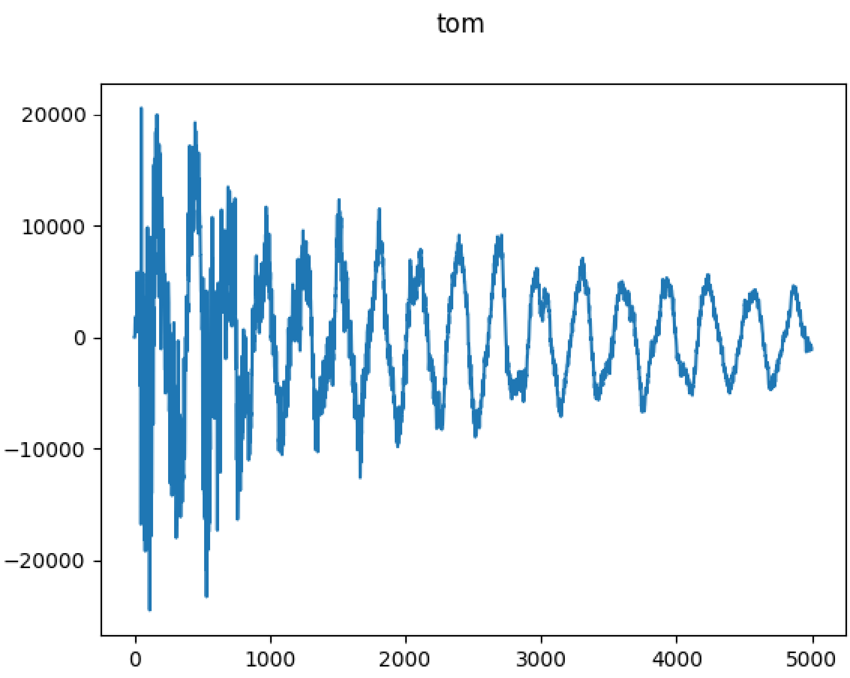

Figure 14. The average waveform of class tom

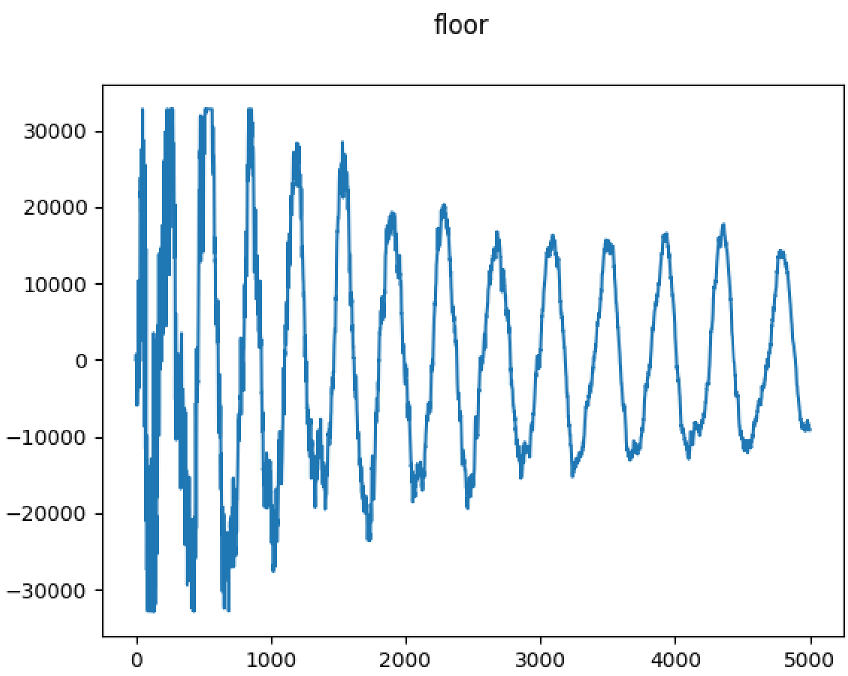

Figure 15. The average waveform of class floor

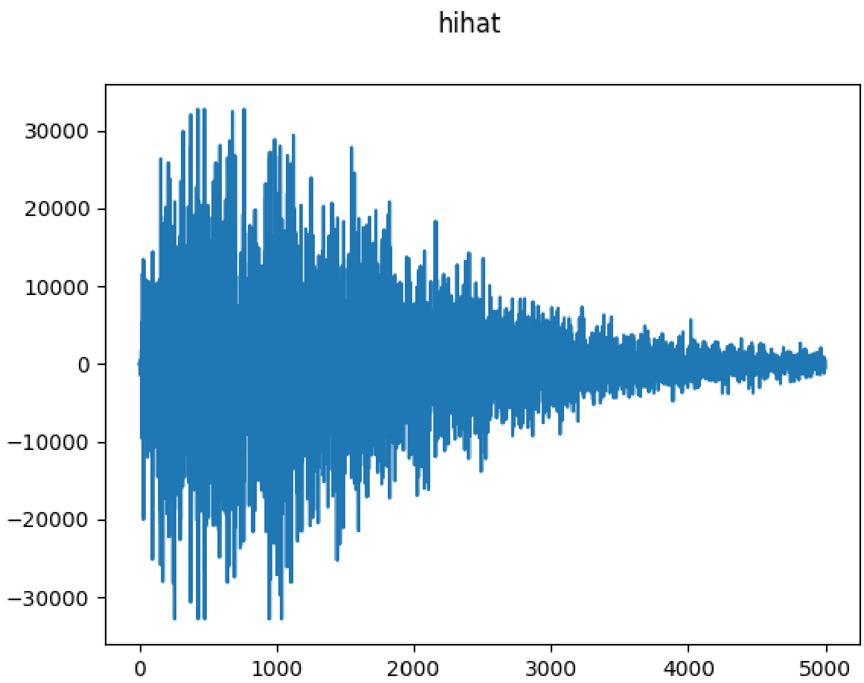

Figure 16. The average waveform of class hihat

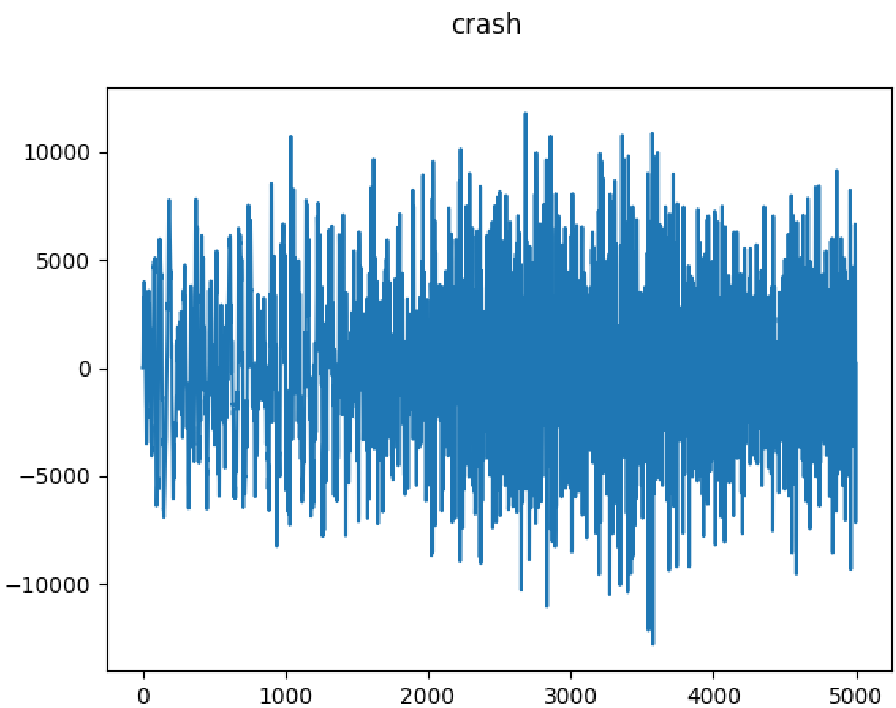

Figure 17. The average waveform of class crash

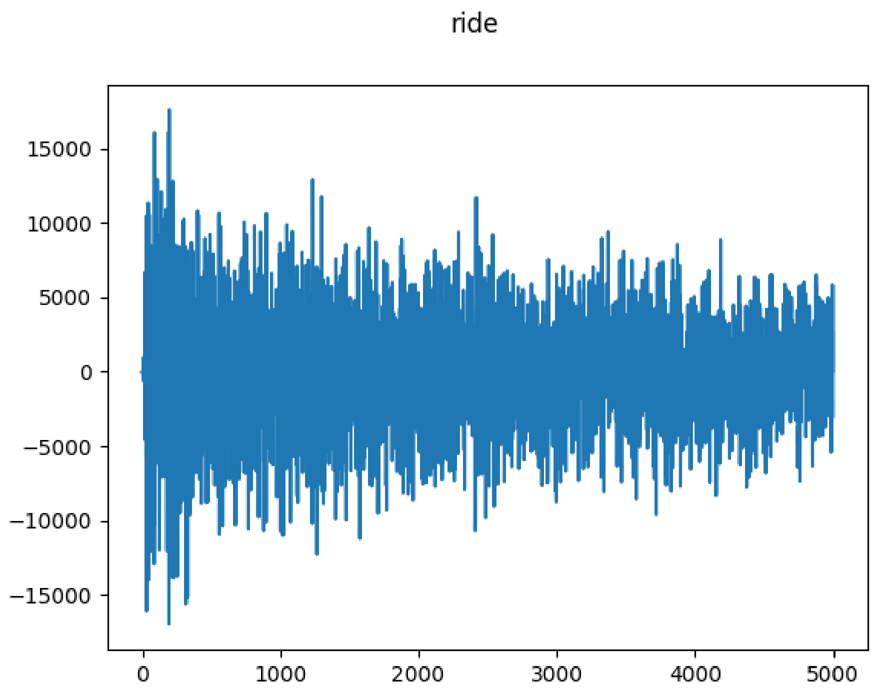

Figure 18. The average waveform of class ride

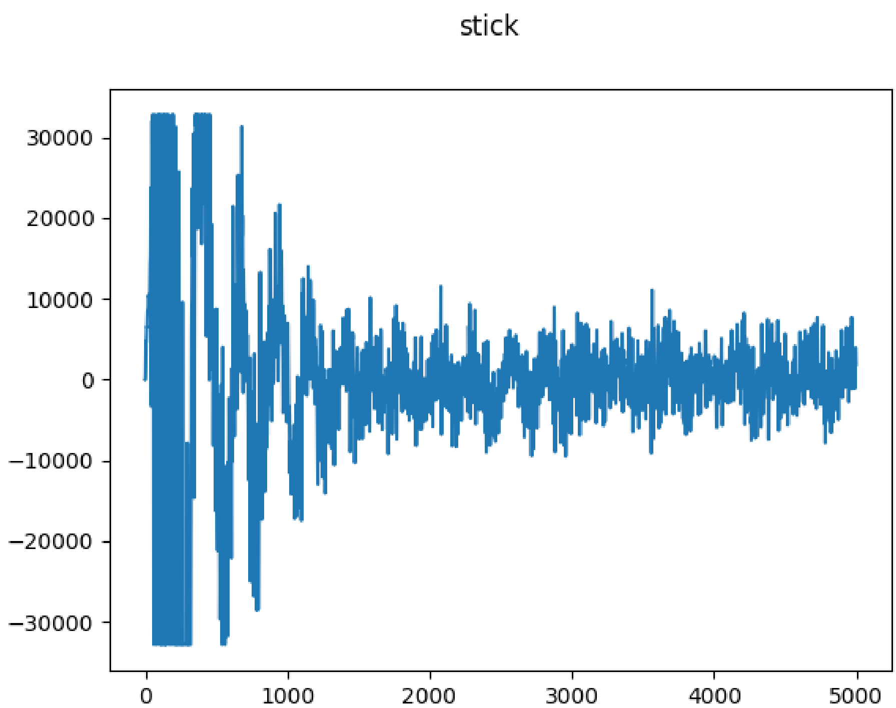

Figure 19. The average waveform of class stick

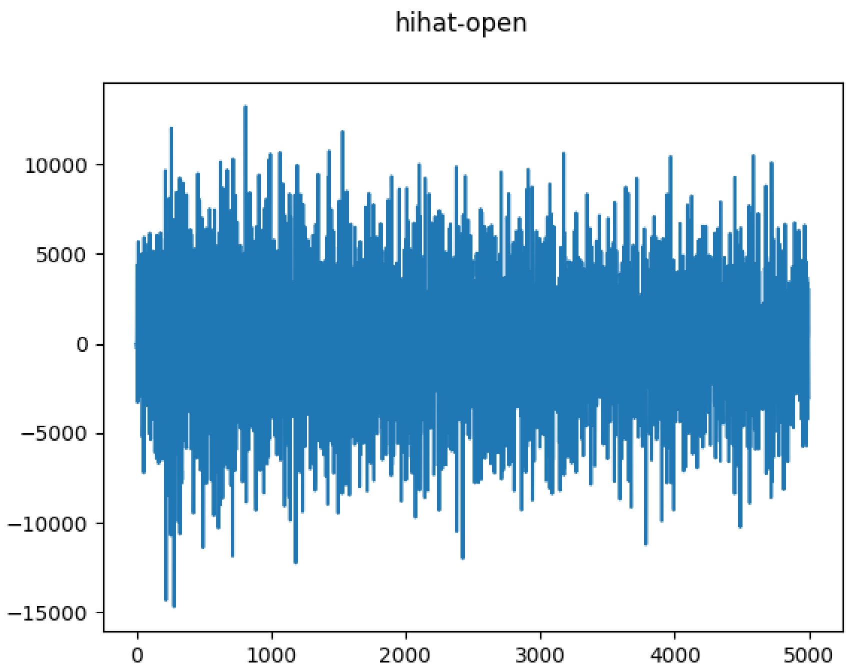

Figure 20. The average waveform of class open-hihat

APPENDIX IV. SAMPLED DRUMS AVERAGE FEATURES

The averaged visualization of each class features

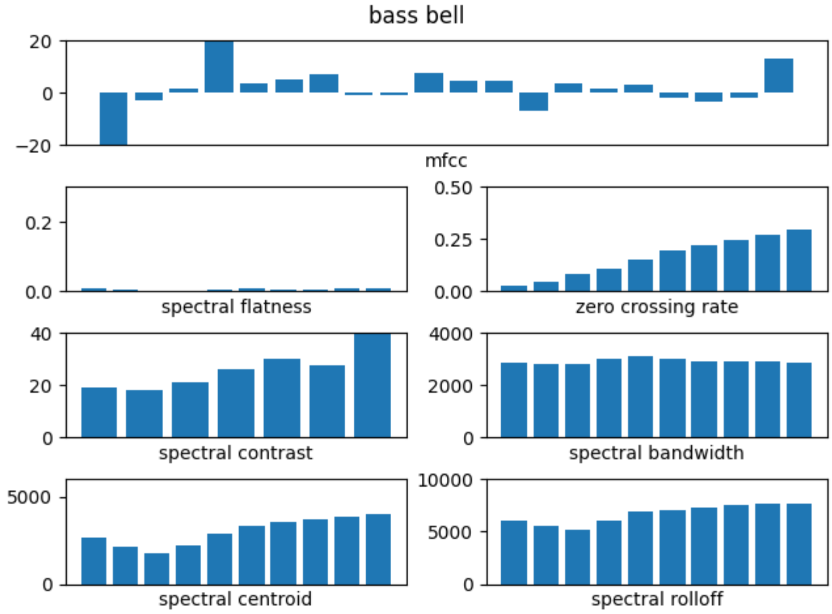

Figure 21. Features of class bass bell

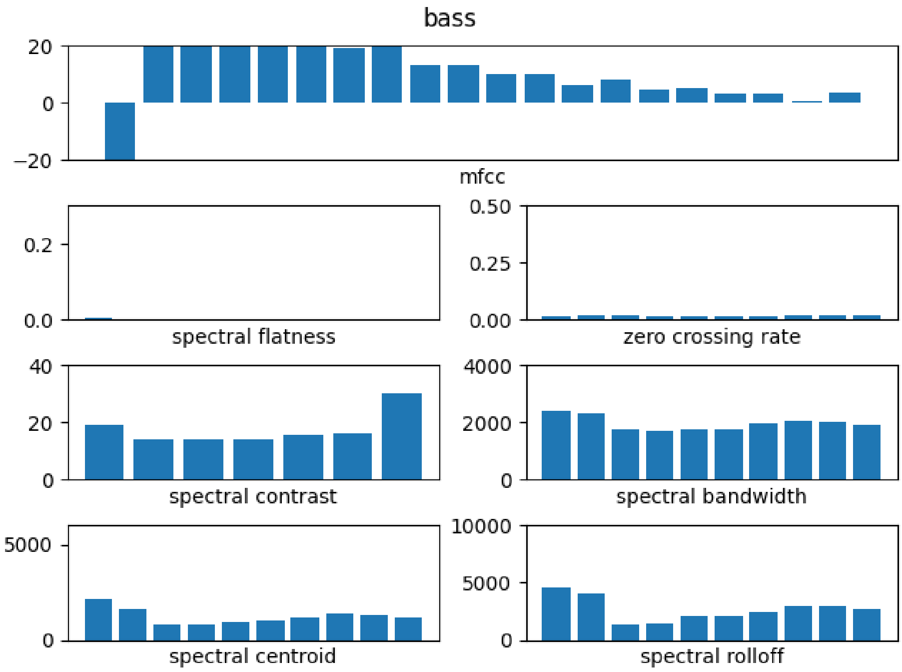

Figure 22. Features of class bass

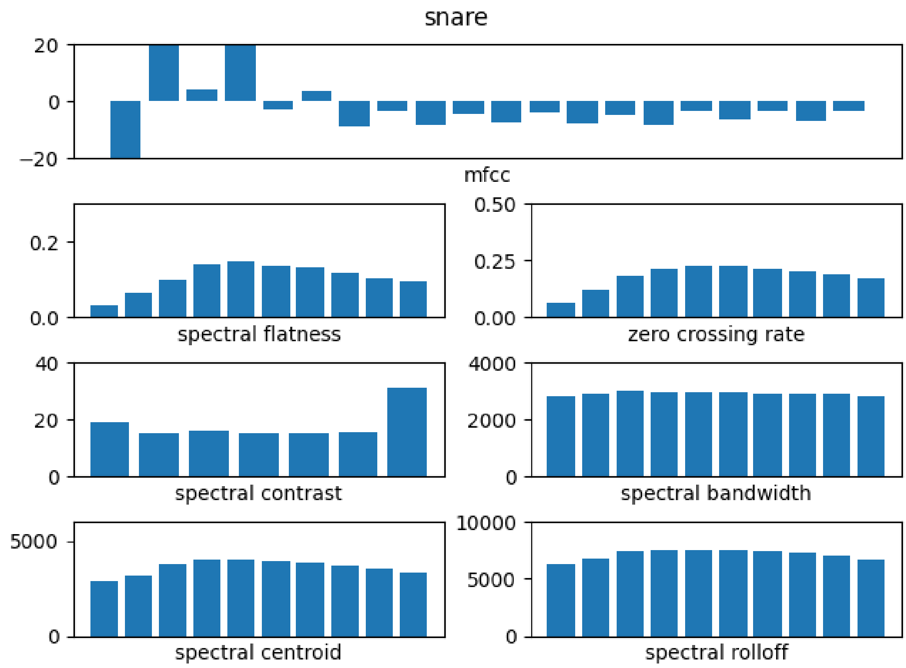

Figure 23. Features of class snare

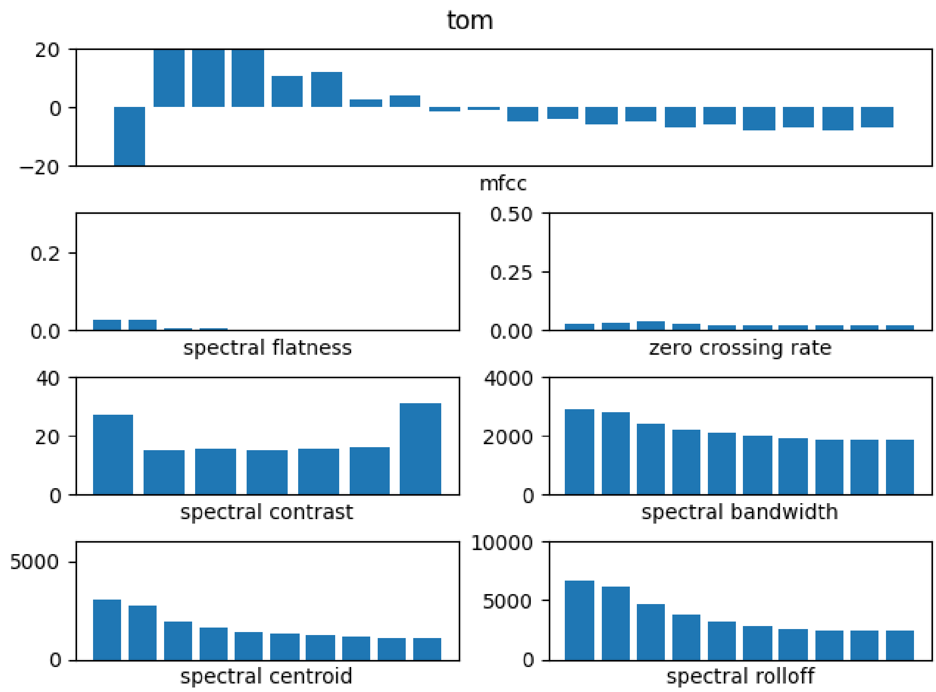

Figure 24. Features of class tom

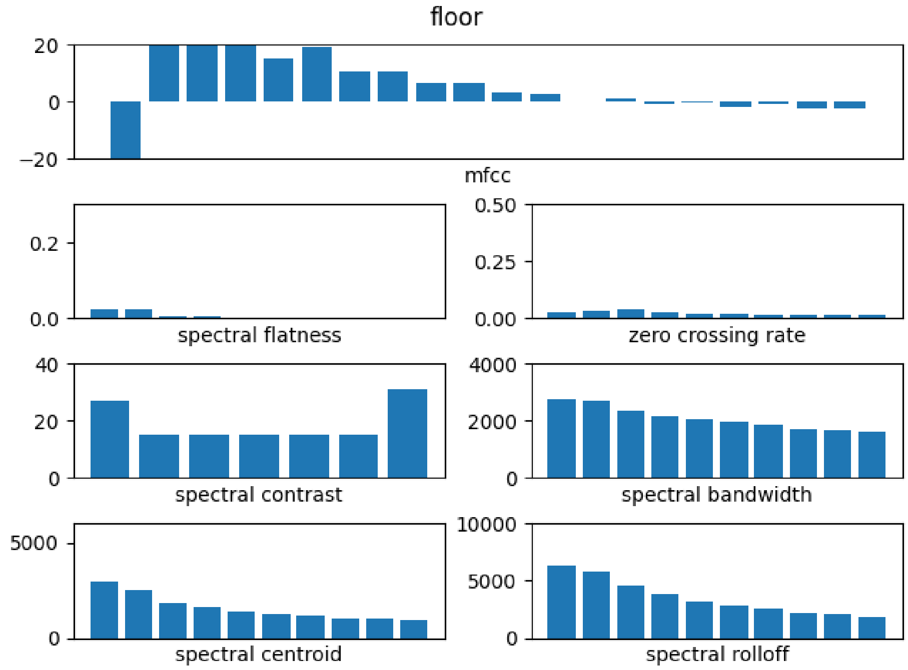

Figure 25. Features of class floor

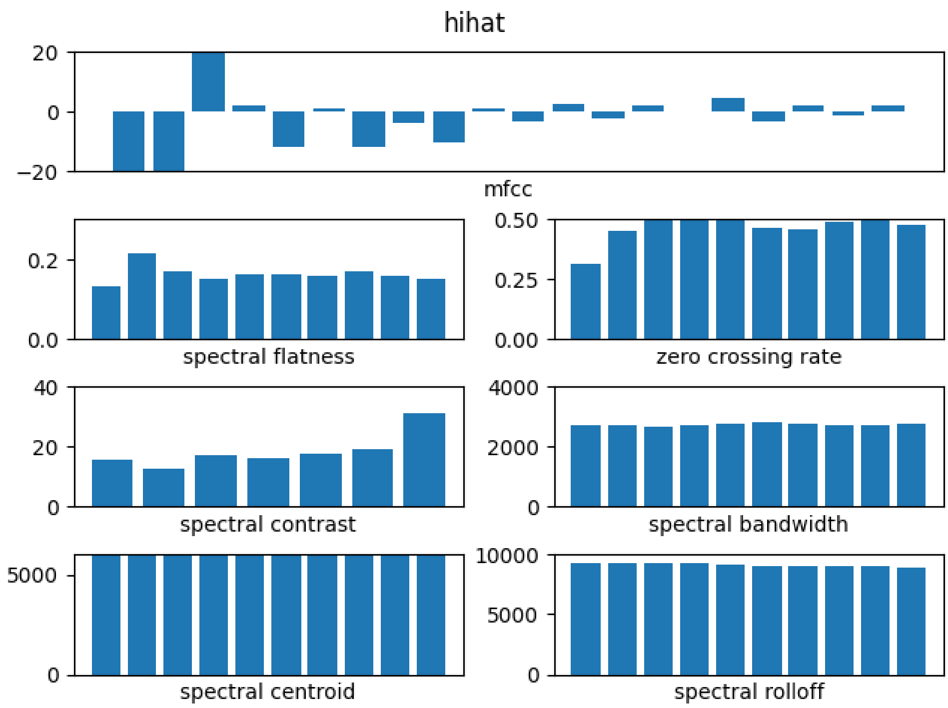

Figure 26. Features of class hihat

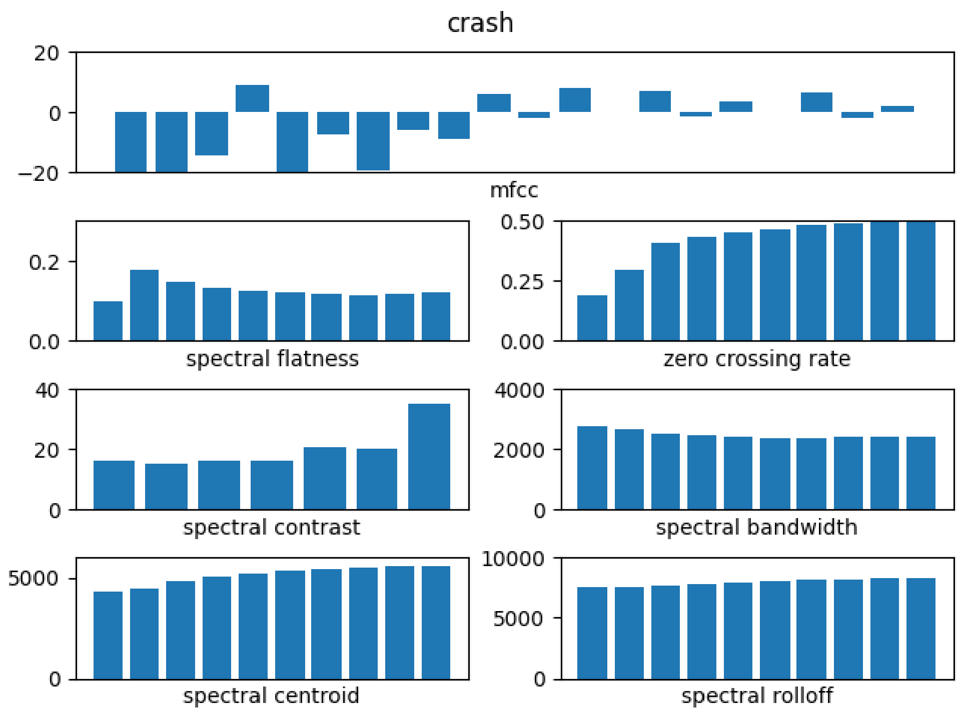

Figure 27. Features of class crash

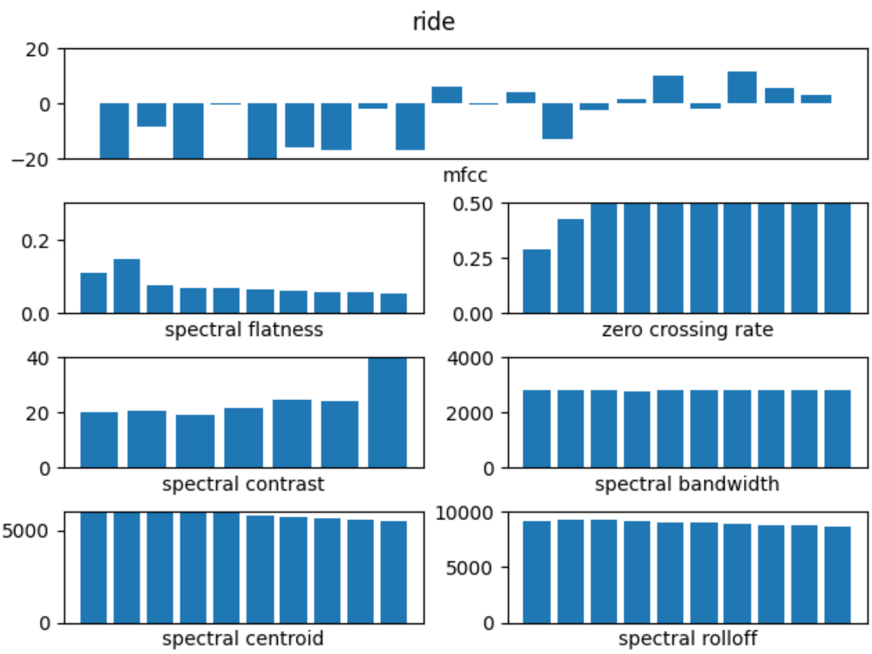

Figure 28. Features of class ride