Automated Drumline Rhythm and Instrument Recognition

Levin Leesemann
Department of Computer Science
University of Iowa
14 MacLean Hall, Iowa City, IA 52242
levin-leesemann@uiowa.edu

Abstract - **Drumlines typically rely on sheet music or some type of non-audio record, however acquiring or producing such a resource can be an expensive and time consuming endeavor. Previous research and projects have shown that instrument and rhythm recognition can be achieved using a variety of approaches, ranging from very complex and sophisticated systems, to machine learning based approaches. This paper aims to demonstrate that the capabilities of neural networks and readily available audio analysis libraries can be used within a system to help transcribe drumline specific audio recordings. Instrument recognition and rhythm analysis are the two main problems when transcribing drumline music, which are approached by using machine learning and a logical approach, respectively. The system shows that a neural network is capable of recognizing drumline instruments from short audio excerpts. It also shows that accurately identifying common rhythms is attainable using a simple system that can be expanded to interpret more complex rhythms.**

Index Terms - Machine Learning, Audio Analysis, Drumline

I. Introduction

Could software aid in dramatically speeding up the tedious music transcription process? Generally speaking, music consists of an underlying time structure, rhythm, melody, harmony, and dynamics. Transcribing music requires an understanding of each of these aspects to produce an accurate transcription. When compared to other instruments, particularly pitched wind and string instruments, drumming is usually distinguishable by its polyphonic nature, focus on rhythmic virtuosity, and absence of melodic choice in playing.

For the purposes of this project the focus is on drumlines, particularly Drum Corps International (DCI)/Winter Guard International (WGI) and high level university/high school corps-style drumlines. These types of drumlines typically consist of multiple drums (snares, basses, and tenors) with each of them having rhythmically intricate/complex parts. Drumlines were selected for this project, since having sheet music is a common tool, and because having a general non-audio record of a piece of music is useful for logistical and sharing purposes.

The goal of this paper is to determine the following: how accurately can software identify both individual drumline instruments and the rhythms that they are playing? To answer this question, the paper first discusses previous work on related subject matter. Next, the methods with their associated design decisions used for this project will be discussed. Finally, this paper will discuss the results of the project, followed by discussion of limitations and future work.

II. Previous Work

Detecting what instrument is playing something and what is being played from an audio file has been shown to be possible in a number of papers. In a melodic/harmonic focussed context [1] demonstrated that the intricate differences in pitch of a piano can be detected using machine learning. [2] showed how using recordings of approximate hummed melodies and rhythms could be used to attain results in a larger system, where recognition was only a small part of it.

A drum set specific application was shown in [3], demonstrating the possibility of a combined system for both instrument and rhythmic detection, specific to drums. Similar to [3], many smaller projects developed by individuals or small teams, such as [4], demonstrate complete basic drumset transcription. Different approaches to specifically rhythm detection have also been shown in [5] and [6], for example.

Building on aspects of the aforementioned work, this project aims to use proven instrument recognition processes with neural networks and a modular/expandable rhythm detection system, to demonstrate how an application that automatically transcribes a drumline recording into sheet music could be created using current software.

III. Methods

Individual drumline instrument parts are commonly broken down into two main components when being analyzed for transcription: the rhythm being played and on what part of the instrument each note is being played (henceforth referred to as “voicings”). In a drumline setting however, multiple instruments will be doing this at once. To account for transcribing multiple instruments in the system, two main processes are applied to a given drumline recording: an instrument recognizer that uses a neural network to determine which instrument(s) are playing a certain note, and a rhythm detector that looks at what each beat can be subdivided into (eighth notes, sixteenth note triplets, etc.). Combining instrument recognition with rhythm detection results in a system that can accurately represent musical transcriptions of drumline recordings.

The following libraries are used for this project: Librosa [7] and SoundFile [8] for audio file processing and handling, Keras [9] for neural network implementation, and NumPy [10] for data handling.

A. *Onset Detection*

The distinct staccato nature of percussion instruments is a large contributor to the unique sound of drums. Notes can clearly be identified due to the strong and short transients at the beginning of each note, which are commonly referred to as onsets.

Onset detection is responsible for detecting when a new note is starting to be played by either one or multiple instruments at once, and is a critical part to both instrument recognition and rhythm detection. Fig. 1 shows a visual example of where onsets occur in an audio file.

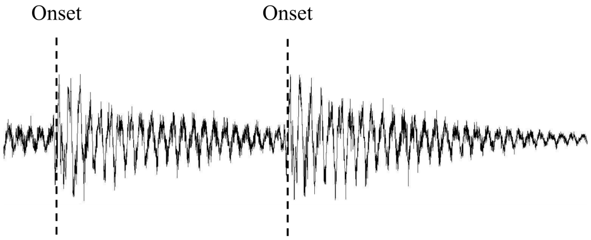

Fig. 1. Waveforms of two subsequent snare drum hits

[7] contains an implementation that can detect onsets well when hyperparameters are tweaked to specific values for drumline instruments. To ensure the precision of the detected onset times, the system passes over each of them and adjusts them as necessary. The adjustment is achieved by creating a window starting 20ms before the detected onset time, and 50ms afterwards. The peak amplitude within that window is then set to be the updated onset time. The aforementioned window start and end time were selected to be large enough to ensure that the true peak of the onset was contained within it, and small enough to not allow the window to contain peaks from different onsets.

B. *Instrument Recognition*

Traditionally, audio is represented in the waveform format. The waveform format is not ideal for analysis due to audio being represented by only a single value at any given point in time. To get good results, the decision was made to use the spectrogram format. The conversion from the waveform format to the spectrogram format was achieved via the short-time Fourier transform (STFT) function included in [7]. Unlike the waveform format, the spectrogram format stores both frequency and amplitude over time. To a person this makes visually identifying what instruments might be present at any given time easier, but it also allows much more information to be stored about the audio at any given point in time as well, a crucial part to achieving successful results with a neural network.

Determining the right amount of data to analyze for each note was based on the realistic upper limit of playing ability of experienced individuals in a drumline setting. Double stroke rolls were considered to be the limit, due to their low interonset interval while maintaining easily distinguishable sounds of notes. Buzz rolls would be the next step up in terms of note density, however they are much closer to a continuous note, and thus were not considered.

Ultimately, the window size of data used to analyze each note was selected to be 50 milliseconds (these windows of data will henceforth be referred to as “chunks”). This chunk size was selected to retain the ability of transcribing audio containing the aforementioned upper limit of playing ability, and is therefore the most amount of data available for each note. The chunk length still contains a small amount of overhead to account for error in onset detection, but is evidently long enough to gather sufficient data for the neural network.

The deep neural network structure used in [1] was adapted for this system. Fig. 3 shows the detailed layout of the neural network.

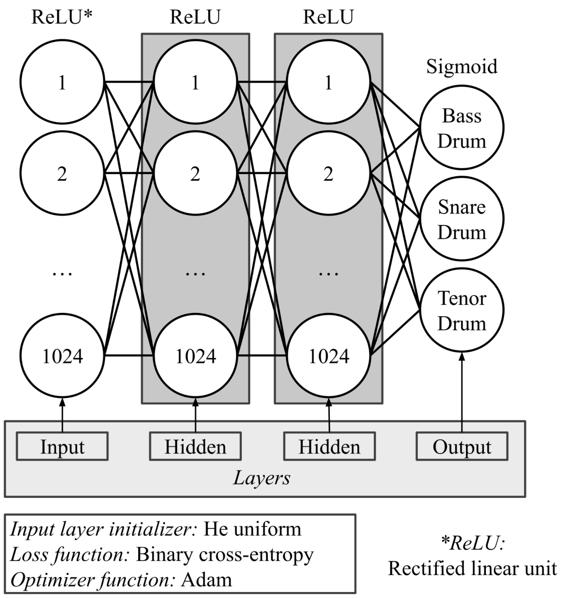

Fig. 3. Neural network architecture

Premature experimentation (Tab. 1) on small datasets led to the decision to use 2 hidden layers. Although 3 hidden layers yielded better individual scores for snare drums than either 1 or 2 hidden layers did, 2 hidden layers were selected due to drumline settings rarely having only one instrument playing at a given time. Section 4.B shows lower single-instrument performance than multi-instrument performance, which could possibly stem from the lower number of hidden layers as well. The neural network model was trained for 10 epochs due to it being a reasonable balance between accuracy, loss, and computation time on the hardware used. Fig. 4 shows how the multi-class f-scores (based on [13]) used in this paper were calculated.

	1 layers	**2 layers**	3 layers
All Snares	0.4	**0.4**	0.69
All Tenors	0.91	**0.907**	0.91
Mixed	0.264	**0.47**	0.35

Tab. 1. F-scores of models with different numbers of hidden layers, trained on small datasets

$$f\text{–}score(class) = 2 \cdot \frac{presion(class) \cdot recall(class)}{precision(class) + recall(class)}$$

Fig. 4. Multi-class f-score formula

The input layer takes in the aforementioned 50 millisecond chunks, which are extracted starting at each onset, and then passed to the STFT function to be converted from the waveform format to the spectrogram format. These formatted chunks are flattened before entering the input layer. Two hidden layers, identical to the input layer in design, then further evaluate the input, before an output containing the most likely instrument(s) present within the chunk is given. This evaluation process is repeated for each chunk.

C. Rhythm Detection

A number of approaches were conceived for rhythm detection, including options using neural networks. To limit the scope of the system, the decision was made to use beat-by-beat evaluation. It is a common way to listen to music, is simple to implement, covers the majority of common rhythms, and is easily expandable (see section 5.B). The implementation of the system handles groupings of one, two, three, and four notes for any given beat. These groupings correspond to quarter, eighth, quarter-note triplets, and sixteenth notes, respectively.

Using the time difference between each onset and the time at which each beat occurs, each onset is assigned to the corresponding beat it falls under. Subsequently, the subdivision for each beat is extracted by seeing under which subdivision grid each of the onsets aligns.

Detecting what subdivision occurs at a given beat is achieved by applying the following process for each subdivision: divide the beat length by the maximum number of notes the subdivision can contain, compare the times calculated with the times of each note, and keep the first subdivision that the notes apply to (if ascending from one to four notes while checking each subdivision, as is done in the system's implementation). The tempo of the recording is needed for this process, and since it is commonly known or relatively easy to determine, it is entered by the user when the system is run.

A numeric example can be used to explain how a subdivision's times are calculated: if a beat length is 0.5 seconds for example (given a beats per minute value of 120) and the beat occurs 20 seconds into the audio file, then the sixteenth note (4 notes per beat) subdivision would have notes occurring at the following times: 20.000, 20.125, 20.250, and 20.375.

Fig. 2 shows a high-level example of how a beat is evaluated using the above outlined method.

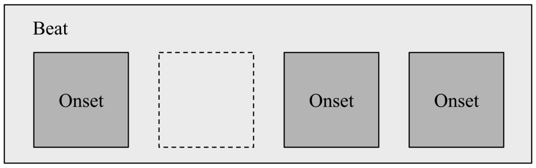

Fig. 2. Visual representation of how onsets fill a sixteenth note subdivision

It is important to note that the beat could have also been broken down into the thirty-second note subdivision (or an even finer subdivision for that matter), however the most sensible approach is to use the simplest subdivision possible.

D. Data Generation

Most pieces of drumline music don't contain each combination of instrument voices, however it is very probable that any combination of instrument voices can occur in a set of drumline audio recordings. To account for this in the system, short chunks (the same ones used in the neural network) from isolated instruments are extracted individually. The number of chunks extracted for a given instrument exists to avoid overfitting to a specific audio recording, and was chosen to be in the ranges shown in Tab. 2.

Instrument	# of chunks extracted	# of unique instrument voices
Snare Drums	Low 10's	1
Tenor Drums	Low 100's	4 - 6
Bass Drums	Low 100's	5 - 8

Tab. 2. Number of audio chunks used per instrument for neural network dataset generation

Isolated instrument chunks are combined with isolated chunks of other instruments in each realistic way possible to create multi-instrument chunks. Due to extensive computation time from file combination and creation, multi-instrument chunks were selected to be limited in the system by a value *n* that determines every *n*th single-instrument chunk that should be used during the combination process. The first chunk is selected randomly between the first and *n*th file, which prevents the neural network from training and testing on the same chunks every time the system is run.

IV. Results

A. *Instrument Recognition*

To measure the main performance of the neural network, the f-score metric was used. For reasonable computation time, an *n* value (described in section 4.D) of 8 was used to generate multi-instrument files, yielding just under 13,000 files with the audio files used. Training was done with a randomized 80:20 (training:test) split. The results (Tab. 4 and Tab. 5) reveal information that likely has some intuitive reasoning behind it.

	Snare Drums	Bass Drums	Tenor Drums
F-score	**0.995**	**0.991**	**0.989**
Precision	0.995	0.990	0.989
Recall	0.995	0.993	0.990
Accuracy	**0.991**	**0.984**	**0.979**

Tab. 4. F-scores for individual instruments

	Singles	Doubles	Triples
Accuracy	**77.04%**	**85.42%**	**98.98%**

Tab. 5. Accuracy for each number of instrument combinations

The lower accuracy of tenors and bass drums hints at their similar timbres leading to mislabeling between the two. Investigation into cases of individual mislabeling revealed that the two are often mistaken for each other, or in the case of multiple instruments being present at once, the chunk being evaluated is thought to have both of the instruments when there is only one, or vice versa.

With an *n* value of 8 being used, the number of multi-instrument files created was cut down from around 3,300,000 to just 13,000. Using more multi-instrument files would likely result in even better outcomes in all of the categories that were evaluated. It should be noted however, that increasing the number of multi-instrument files without increasing the number of single-instrument files will likely lead to the neural network mislabeling more single-instrument chunks as multi-instrument chunks.

B. *Rhythm Detection*

To evaluate each of the subdivisions that were implemented for this system, a short recording produced in [12], that contained variations of each of the subdivisions, was used (Fig. 5). The variations vary in how many notes they have, where each of the notes are located within the subdivision, and whether a downbeat is present.

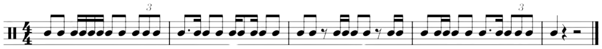

Fig. 5. Musical notation of the snare drum recording

For the recording in question, the system's determinations (Tab. 3) were accurate, and clearly demonstrate how the beat-by-beat approach of evaluation logically works.

Having a visual representation of how the system evaluates each rhythm shows how additional subdivisions can easily be added. It also gives an intuition as to how adding support for rhythmic groupings longer or shorter than a beat can be achieved by adapting some of the same subdivisions used in the single beat subdivision recognition to those different grouping lengths.

Beat	Subdivision	Detected Notes
1	Eighth	1, 2
2	Sixteenth	1, 2, 3, 4
3	Eighth	1, 2
4	Eighth-Triplet	1, 2, 3
5	Sixteenth	1, 4
6	Eighth	1, 2
7	Sixteenth	1, 2, 4
8	Eighth	1, 2
9	Eighth	1, 2
10	Sixteenth	3, 4
11	Eighth	1, 2
12	Sixteenth	3, 4
13	Sixteenth	1, 3, 4
14	Eighth	1, 2
15	Sixteenth	1, 4
16	Eighth-Triplet	1, 2, 3
17	Quarter	1

Tab. 3. Extracted subdivision for each beat, including which notes of the subdivision were found to be matched

V. Discussion

A. Limitations

As discussed in section 1, DCI/WGI-style (also known as corps-style) drumline recordings are the main focus of this system. On one hand, this helps limit inconsistent time-keeping, “dirt” (inconsistencies in rhythm across multiple players), and irregular drum sounds/tuning schemes. On the other hand it doesn’t allow for the system to be used for show-style drumming. The latter style focuses more on entertainment, leaving out the more complex rhythms, and being more tolerant to dirt.

Pitch differences between different sized tenors and bass drums vary from one audio recording to another, which can lead to it being very difficult for an accurate transcription to be made without an accompanying video of each instrument, or recordings of each instrument voice for a given drumline. To limit the scope of this system, all notes that contain tenor or bass drums are simply grouped into a single tenor or bass drum label.

Cymbals are part of certain drumlines as well, although not as common as snare, bass, and tenor drums. Due to cymbals being substantially different in timbre to drums (crashes, chokes, taps, zings, sizzles, etc.) and their relative rarity, the decision was made to omit them from this system. To account for the different sounds cymbals can make, a different approach would need to be taken to this system; likely one that is more similar to traditional instrument recognition that handles held notes and releases, such as is the case with many melodic instruments.

B. Future Work

To allow the system to recognize the individual voices of an instrument, an approach of progressively mapping the pitches heard in a recording to a guessed structure of the recorded drumline could be taken. Having a recording of each individual drum and considering them individual instruments within this system would likely be more reliable, but not be feasible in all situations. A cross between both of the aforementioned approaches, in which common voicings for each instrument have already been trained on, could be taken, with the model adapting to the voicings in a recording over time to get better results, specific to that recording.

Parsing the information received from this system into a tool such as LilyPond [11] would result in a system that could simply take in a drumline audio file and return a well-formatted piece of sheet music. The information received from the system could also be parsed into a MIDI format so that systems such as MuseScore [12] could import a generated MIDI file and modify, correct, or further add to the transcription generated by the system. To achieve this, the current system would need to be automated to the point where all a user would have to do is run the system while passing in a drumline audio file and the associated tempo. The output would simply be a piece of sheet music or MIDI file.

The number of rhythmic combinations is endless, due to factors such as subdivision group length (the system only accounts for lengths of 1 beat for instance), partial rhythms, nested rhythms, and ratio rhythms. A system accounting for each possibility would need to be very substantial. Building on the current system however, adding individual subsystems to address each different rhythmical concept could be one approach to allow the system to understand more complex rhythms. This approach would result in a very modular design in which each subsystem could be individually adjusted and evaluated.

It should be noted however, that due to the nature of drumline music, the endless combinations of rhythms are limited to maintain playability while marching. Drumlines generally march or “mark time” (quasi marching in place) to a constant pulse, typically the quarter note, and are only rarely at a standstill. This leads drumline music to generally only contain subdivisions of 1, 2, 3, 4, 6, and 8. More advanced drumlines will also include subdivisions of 5, 7, and 9, although occurrences of them are rare. These subdivisions lend themselves well to the system described in this paper, due to the simplicity with which they can be detected and implemented.

VI. Conclusion

The results of the main processes suggest that current software is capable of performing two of the main tasks of transcription specific to drumlines: rhythm detection and instrument recognition. In retrospect, changes to the parameters of the neural network, onset detection approach, and rhythm detection could likely yield better results. Adjusting the current processes and further developing them along with new processes into one fully automated system seems to be an attainable goal. The processes that make up the system in its current state exist almost completely separately, apart from shared files, data structures, and a few key functions.

REFERENCES

[1] L. Li, I. Ni, and L. Yang, “Music Transcription Using Deep Learning,” Stanford University, Stanford, CA, 2017

[2] N. Mostafa, Y. Wan, U. Amitabh, and P. Fung, “A Machine Learning based Music Retrieval and Recommendation System.” [Online]. Available: https://aclanthology.org/L16-1312.pdf. [Accessed: 04-May-2022].

[3] C.-W. Wu, C. Dittmar, C. Southall, R. Vog, G. Widmer, J. Hockman, M. Muller, and A. Lerch, “A Review of AutomaticDrum Transcription” [Online]. Available: https://core.ac.uk/download/225490566.pdf. [Accessed: 04-May-2022].

[4] Southall, C. (n.d.). Carlsouthall/adtlib: Automated Drum Transcription Library. GitHub. Retrieved April 16, 2022, from https://github.com/CarlSouthall/ADTLib

[5] C. Raphael, “Automated Rhythm Transcription.” [Online]. Available: https://ismir2001.ismir.net/pdf/raphael.pdf. [Accessed: 04-May-2022].

[6] A. Elowsson, “Tempo-Invariant Processing of Rhythm with Convolutional Neural Networks.” [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1804/1804.08661.pdf. [Accessed: 04-May-2022].

[7] “Librosa,” *librosa*. [Online]. Available: https://librosa.org/doc/latest/index.html#. [Accessed: 19-Mar-2022].

[8] “Soundfile,” SoundFile. [Online]. Available: https://pysoundfile.readthedocs.io/en/latest/. [Accessed: 19-Mar-2022].

[9] K. Team, “Simple. flexible. powerful.,” *Keras*. [Online]. Available: https://keras.io/. [Accessed: 19-Mar-2022].

[10] NumPy. [Online]. Available: https://numpy.org/. [Accessed: 19-Mar-2022].

[11] “Music notation for everyone,” *LilyPond*. [Online]. Available: http://lilypond.org/. [Accessed: 20-Mar-2022].

[12] “Create, play and print beautiful sheet music,” *MuseScore*. [Online]. Available: https://musescore.org/en. [Accessed: 20-Mar-2022].

[13] B. Shmueli, “Multi-class metrics made simple, part II: The F1-score,” *Towards Data Science*, 03-Jul-2020. [Online]. Available: https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1. [Accessed: 12-Apr-2022].