Real-Time Identification of Common and Extended Musical Chords from MIDI Signals using Neural Networks

Introduction

Chords are collections of two or more musical notes, often played together, and are arranged in such a way that they follow the so-called "rules of harmony" (Leino, Brattico, Tervaniemi, & Vurst, 2007). These rules are recognized by humans as a response in the brain that is triggered when they are violated (Leino *et al.*, 2007). Because of this, chords are fundamental to the harmonic integrity of any musical work.

A musical chord is commonly identified by three parameters: root note, chord type, and inversion. The root note serves as a reference point for the other notes which are played in the chord. These other notes are determined by the chord type. The inversion tells which note in the chord is played lowest. For example, a C major chord, 0^{th} inversion has "C" as the root note, and "major" as the chord type. A major chord type includes the 1^{st} (root), major 3^{rd}, and 5^{th} harmonics of the typical Western major scale, and the 0^{th} inversion says that the root note is the bottom note. Thus, the notes of the 0^{th} inversion of a C major chord are C (1^{st}), E (major 3^{rd}), and G (5^{th}), in that order.

Statement of the Problem

Humphrey, Bello, and Cho (n.d.) state that "the general music learning public places a high demand on chord-based representations of popular music" (par. 1). However, complete and accurate determination of these chords by hearing requires the use of both absolute and relative pitch, because chords utilize both an absolute reference point (root note) and a relative configuration of harmonies (chord type).

Absolute pitch is expressed when one can identify a musical note by hearing it, while relative pitch is shown when one can recognize the distances between musical notes (Zatorre, Perry, Beckett, Westbury, & Evans, 1998). While "most trained musicians" (Zatorre *et al.*, 1998) exhibit a mastery of the latter, few of them have absolute pitch.

Absolute pitch is expressed in a low percentage of the human population and acquired through a combination of favorable genes and music training at a young age (Baharloo, Service, Risch, Gitschier, & Freimer, 2000). Therefore, complete chord identification is a rare skill found in those

with mastery of both absolute and relative pitch, even though chords play an important role in any musical work. An algorithm that automatically identifies chords from individual notes would be a first step towards addressing this problem.

Objectives of the Study

This study aims to develop a neural network that quickly and correctly identifies common and extended one-root musical chords formed by playing more than two notes on a MIDI input device. Specifically, the neural network must identify common & extended chords and respond quickly enough to be used in live performance (Stark & Plumbley, 2009). The program must be implemented in programming languages that have MIDI input-output libraries such as *pyrtmidi* (Kidd, 2017) for Python and *rtmidi* (thestk, 2017) for C++ and neural network libraries such as *hebel* (Bretschneider, 2017) to facilitate ease of coding.

Significance of the Study

Such algorithms are used in the field of music education, where a low proportion of music students have absolute pitch (Gregersen, Kowalsky, Kohn, & Marvin, 1999) despite their demand for chordal representations of music (Humphrey, Bello, & Cho, n.d., par. 1). These algorithms allow said students to learn to *identify the chords they are playing* more quickly and accurately, and help them develop their senses of relative and absolute pitch. They are also used in situations when musicians need to verify the chords they are playing for correctness, which usually happens when they are learning or composing a musical piece.

While automatic chord identification programs exist, they are either classical algorithm implementations that run on audio inputs in real-time (Fujishima, 1999; Stark & Plumbley, 2009), or neural network implementations that do not aim to run in real-time and do not include chords outside of major and minor triads (Osmalskyj, Embrechts, Piérard, & Van Droogenbroeck, 2012; Perera & Kodithuwakku, 2005; Zhou & Lerch, 2015). Including other chords such as 7ths and extended chords would allow identification of chords from more complex styles of music, such as jazz. *The utilization of* ***neural networks*** *to identify* ***extended chords*** *from* ***MIDI signals*** *in* ***real-time*** *is largely unexplored and would provide useful data for future research.*

Scope and Limitation

The study aims to develop a neural network that will quickly and accurately identify chords of only these types (Stark & Plumbley, 2009):

1. Basic triads (major, minor, augmented, diminished, sus2, sus4)
2. 7ths (dominant 7, major 7, minor 7, augmented 7, diminished 7, half diminished, suspended variants)
3. 9ths (dominant 9, major 9, minor 9, augmented 9, diminished 9, suspended variants)
4. 11ths (dominant 11, major 11, minor 11, augmented 11, diminished 11, suspended variants)

Only 0^{th} inversions of these chords will be tested against the neural network.

Research Design

Experimental Units and details to ensure local control

Since the study involves the development of a neural network, each experimental unit would be defined as the first attempt that a neural network (after several training cycles to be determined by future methods) guessed a certain chord. The evaluated runtimes will then be sorted during data collection by *input chord type*.

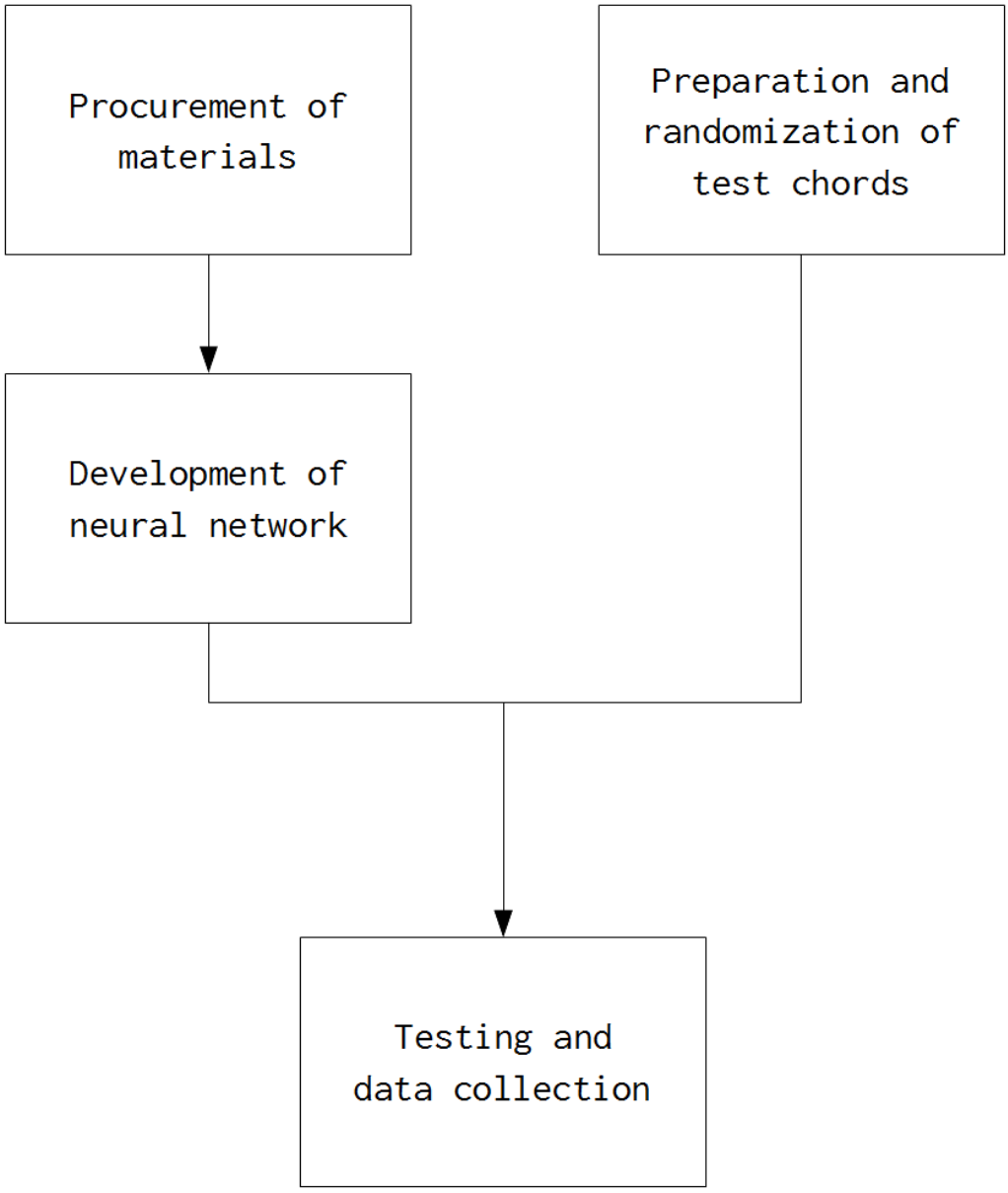
Treatments

Each treatment is a specific chord with its own root note. A fixed number of chords will be randomly selected for every chord type by randomizing said root note. The chord types to be used are those found in the Scope and Limitation of this paper.

Allocation of treatments to EUs

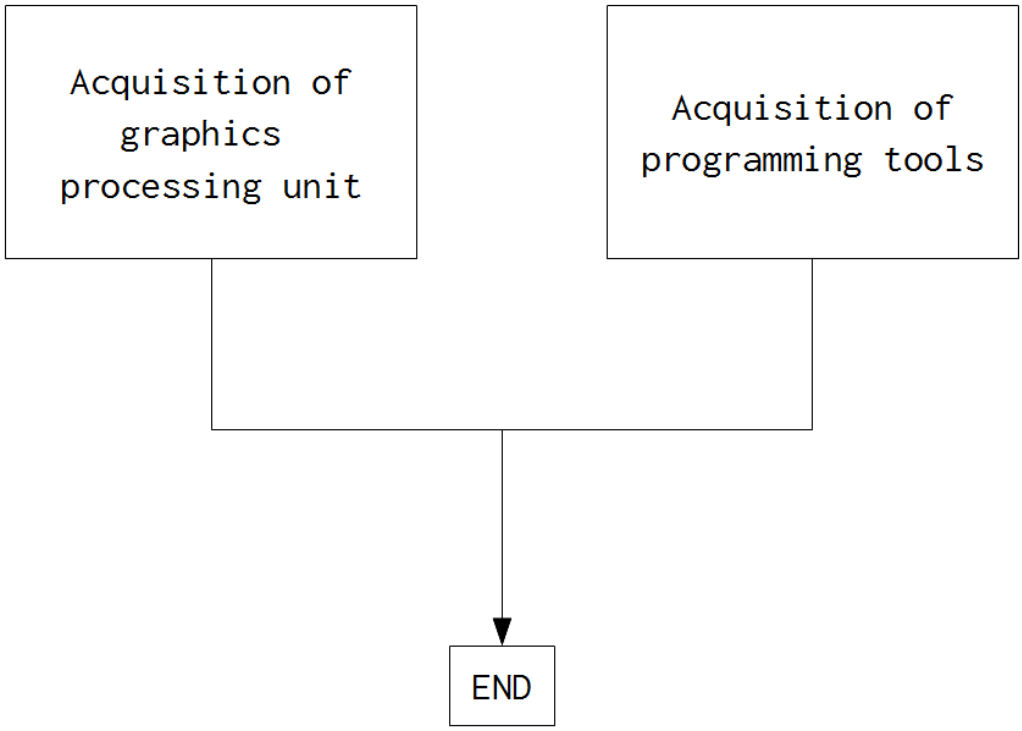
Each chord to test the neural network through will be randomized by randomly obtaining the root note (C, C#, D, D#, E, F, F#, G, G#, A, A#, B), as mentioned in *Treatments*.

Level 0 diagram



Level 1 diagrams

0 PROCUREMENT OF MATERIALS



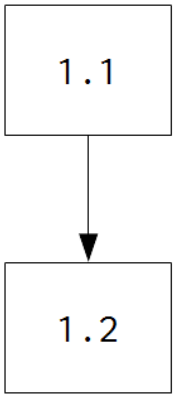
0.1 Acquisition of graphics processing unit (Nickolls, Buck, Garland, & Skadron, 2008)

A GPU or graphics processing unit is useful for parallel computing applications such as neural network simulations and training. Due to its high price, this step will ***only be carried out if*** *the GPU owned by the researchers (nVidia GeForce GTX 960 2GB) is found to be inadequate. Acquisition can be through rental or purchase, whichever is more accessible in terms of pricing and availability.*

0.2 Acquisition of programming tools (Bretschneider, 2017; Kidd, 2017; Nickolls *et al.*, 2008; thestk, 2017)

These include libraries that allow MIDI input and output, neural network simulations, and GPU acceleration.

1 DEVELOPMENT OF NEURAL NETWORK



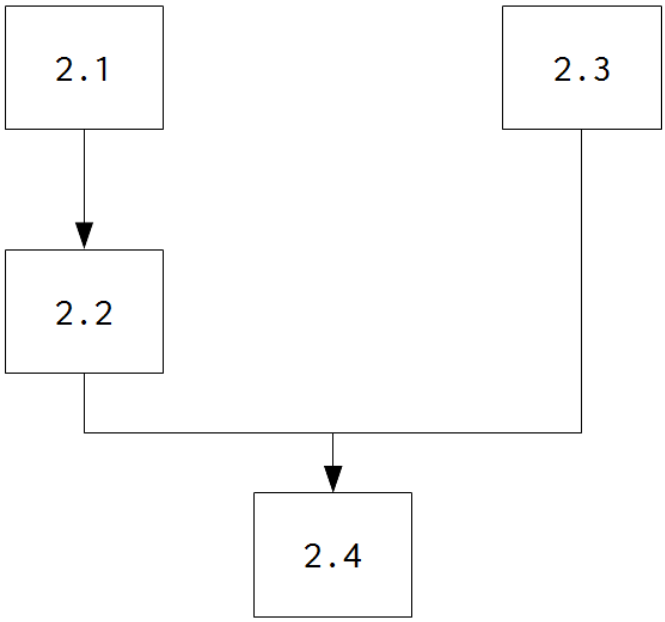
1.1 Neural network and MIDI interfacing setup (Bretschneider, 2017; Kidd, 2017; thestk, 2017)

This involves the use of a programming language and the appropriate libraries to set up a neural network and facilitate MIDI input to the network.

1.2 Neural network training (Perera & Kodithuwakku, 2005; Osmalskyj *et al.*, 2012; Zhou & Lerch, 2015)

Using a method to be determined, the neural network will be trained to identify chords from MIDI input signals.

2 PREPARATION & RANDOMIZATION OF TEST CHORDS



2.1 Assignment of test chord parameters (Stark & Plumbley, 2009) to arbitrary numbers

Both root note and inversion number will be assigned arbitrary numbers to facilitate selection by the RNG in step 2.4.

2.2 Determination of sample size

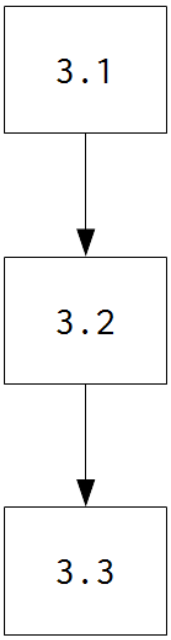
The appropriate sample size will be used to minimize bias and testing duration.

2.3 Acquisition of random number generator

2.4 Stratified random sampling of chords

For every chord type, a root note and inversion number will be randomly selected using the random number generator acquired in step 2.3. Each chord will be run through the neural network and the data will be recorded in step 3.2.

3 TESTING AND DATA COLLECTION



3.1 Neural network testing on randomly chosen chords

To be tested are response time (the time taken by the neural network to generate an output) and chord identification accuracy. The proper testing methods are to be determined.

3.2 Data logging

The chord notation outputs by the program will be collected alongside their respective input chords for analysis in step 3.3.

3.3 Data analysis

The outputs and inputs in step 3.2 will be compared and the appropriate statistical analysis will be applied.

References

Baharloo, S., Service, S., Risch, N., Gitschier, J. & Freimer, N. (2000). Familial aggregation of absolute pitch. *American Journal of Human Genetics, 67,* 755-758. doi:10.1086/303057.

Bretschneider, H. (2017). *hebel: GPU-accelerated deep learning library in Python* [GitHub repository]. Retrieved September 6, 2017, from https://github.com/hannes-brt/hebel.

Brown, R. (2016, July 12). *Rob Brown's answer to Do all electric pianos/keyboards have MIDI?* Retrieved August 24, 2017, from https://www.quora.com/Do-all-electric-pianos-keyboards-have-MIDI.

Fujishima, T. (1999). *Realtime chord recognition of musical sound: A system using common Lisp music.* Retrieved from http://www.music.mcgill.ca/~jason/mumt621/papers5/fujishima_1999.pdf.

Gregersen, P., Kowalsky, E., Kohn, N., & Marvin, E. (1999). Absolute pitch: prevalence, ethnic variation, and estimation of the genetic component [Letter to the Editor]. *American Journal of Human Genetics, 65,* 911-913. doi:10.1086/302541.

Humphrey, E., Bello, J. P., & Cho, T. (n.d.). Chord Recognition. Retrieved from http://steinhardt.nyu.edu/marl/research/chord_recognition.

Kidd, P. (2017). *pyrtmidi: Real-time MIDI I/O for Python* [GitHub repository]. Retrieved August 23, 2017, from https://github.com/patrickkidd/pyrtmidi.

Leino, S., Brattico, E., Tervaniemi, M., & Vurst, P. (2007). Representation of harmony rules in the human brain: Further evidence from event-related potentials. *Brain Research, 1142,* 169-177. doi:10.1016/j.brainres.2007.01.049.

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with CUDA. *ACM Queue, 6*(2), 40-53.

Osmalskyj, J., Embrechts, J-J., Piérard, S., & Van Droogenbroeck, M. (2012, May 9). *Neural networks for musical chords recognition.* Retrieved at http://jim.afim-asso.org/jim12/pdf/jim2012_08_p_osmalskyj.pdf.

Perera, N., & Kodithuwakku, S. R. (2005, December 15). Music chord recognition using artificial neural networks. *1st Proceedings of the International Conference on Information and Automation*, 304-308

Stark, A. M., & Plumbley, M. D. (2009). *Real-time chord recognition for live performance* [PDF file]. Retrieved at https://www.eecs.qmul.ac.uk/~markp/2009/StarkPlumbley09-icmc.pdf.

thestk (2017). *rtmidi: A set of C++ classes that provide a common API for realtime MIDI input/output across Linux (ALSA & JACK), Macintosh OS X (CoreMIDI) and Windows (Multimedia)* [GitHub repository]. Retrieved August 23, 2017, from https://github.com/thestk/rtmidi.

Zatorre, R. J., Perry, D. W., Beckett, C. A., Westbury, C. F., & Evans, A. C. (1998). Functional anatomy of musical processing in listeners with absolute pitch and relative pitch. *Proceedings of the National Academy of Sciences, 95*, 3172-3177. Retrieved at http://www.pnas.org/content/95/6/3172.full.

Zhou, X., & Lerch, A. (2015). Chord detection using deep learning. *16th International Society for Music Information Retrieval Conference*, 52-58. Retrieved at http://ismir2015.uma.es/articles/96_Paper.pdf.