DOCTOR OF PHILOSOPHY

# E3MSD - a new music information retrieval architecture for an original music identifier

Li, Hanchao

*Award date:*
2020

*Awarding institution:*
Coventry University

# E$^3$MSD – A New Music Information Retrieval Architecture for an Original Music Identifier

By

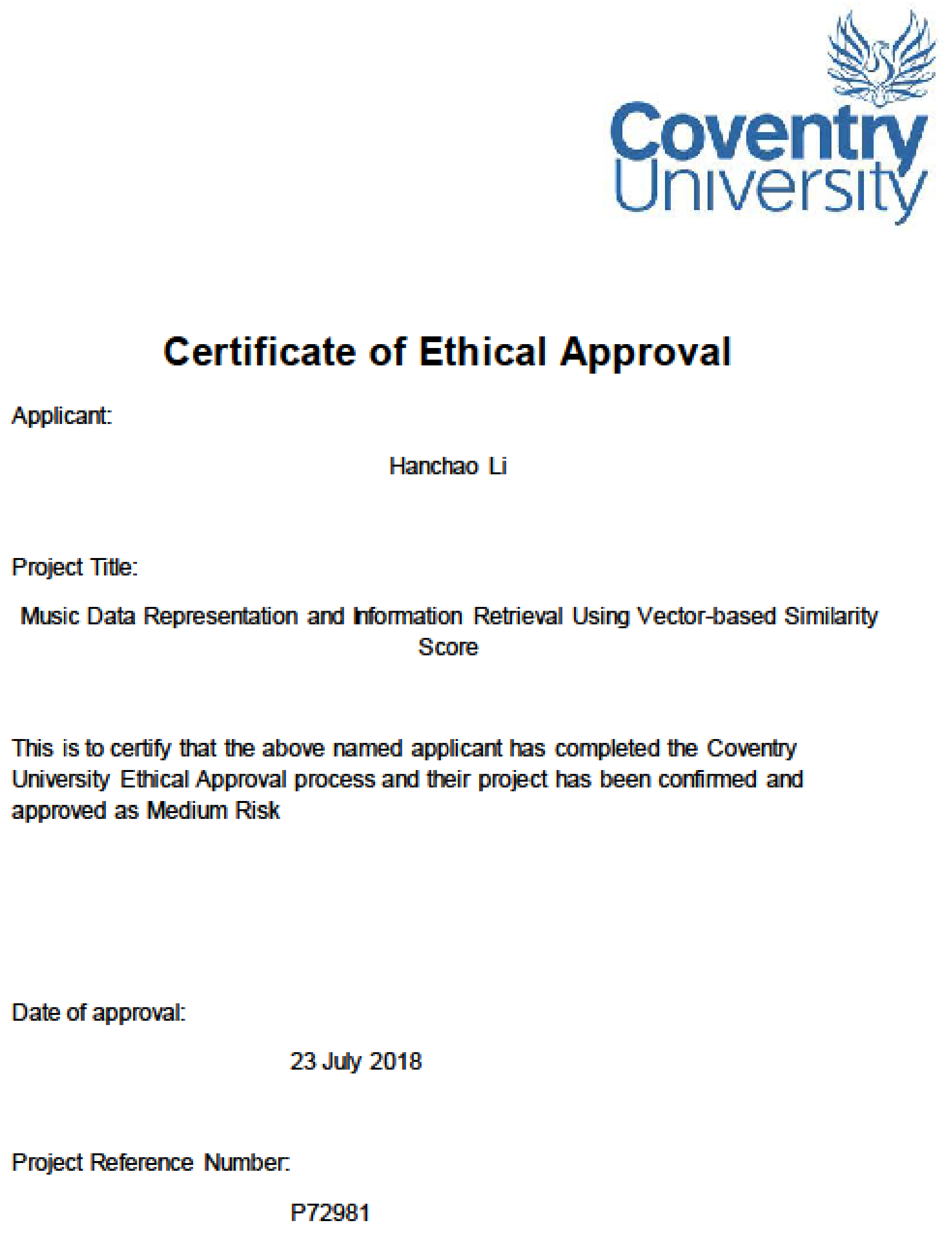
Hanchao Li

May 2020

Coventry University

***A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy***

# Ethical Approval



## Certificate of Ethical Approval

Applicant:

Hanchao Li

Project Title:

Music Data Representation and Information Retrieval Using Vector-based Similarity Score

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Medium Risk

Date of approval:

23 July 2018

Project Reference Number:

P72981

# Abstract

Nowadays, more people tend to use the Internet to search, listen, purchase, download and share music. Existing Music Information Retrieval (MIR) systems were either audio-based or symbolic-based. Audio-based MIR systems were based on various audio formats, for example, MP3. These formats can represent continuous sound waves well but are limited in illustrating the content flow of the music melody and generating large files for high fidelity music. In addition, audio-based MIR systems use the audio fingerprints to find the exact music pieces, but they have difficulty in finding variances. On the other hand, symbolic-based formats have advantages in effectively representing the content of the music with small files, and in facilitating music pattern identifications, but they are not suitable for Electronic Music (EM), which considers a continuous sound wave set. This is because the symbolic-based represention limit themselves to discrete set modelling so that they may introduce extra dummy notes to EM, which can affect the main melody flow and lead to the increase of errors in symbolic-based retrieving including identifying the origins from its variations. As a consequence of these, people have been getting unsatisfied results from both audio and symbolic based music search engines, as well as find it difficult to carry out music plagiarism checks. Therefore, we need to find a new way to describe, model and analyse music.

In this project, we aim to retain those advantages from both audio and symbolic sides while address their shortcomings, as briefly described above, by proposing a new architecture named E$^3$MSD (Expressive, Efficient, and Extendable Music Similarity Detection). There are two contributions for E$^3$MSD.

The first contribution is a new data model that describes the music information using both the Music Definition Language (MDL) and the Music Manipulation Language (MML), which can effectively and efficiently encode and represent the music. For evaluation, we have tested the MDL&MML from the perspectives of music storage and music representation. In terms of storage efficiency, the required storage of a sampled audio encoded by the proposed coding scheme is smaller than other popular audio-based forms. More precisely, a melody, with approximately 316 KB of the file size using the MP3 format, only requires 9 KB disk space when using the MDL&MML format. In terms of music expressiveness, the proposed symbolic-based representation can model various timbre using less storage space without sacrificing the quality. Finally, E$^3$MSD includes the automatic generation of MDL&MML file from the audio soundwaves. The derived MDL&MML file shares around 94% melodic and 100% rhythmic accuracy with manually generated one.

The second contribution is the development of a hybrid mechanism on the proposed musical data model, named MUsic Classification And Similarity Measurement (MUCASM), which combines contour, rhythm and audio fingerprints. This method features a modified reinforcement-based ensemble learning classification mechanism, which includes a decision tree that maps variations of music pieces to their corresponding originals, with variation types as their attributes, for example, rhythm variation. The experimental results show a stable accuracy of 84% without taking into account the types of variations, and 96% by using our proposed ensemble learning.

E$^3$MSD can be extended to study its potentials in improving the performance of existing music search engines, building music version of plagiarism tools, and even generating remixes automatically based on the similarity scores.

# Acknowledgements

This PhD project is 100% self-designed project, where I thought the idea around my Year 3 of Undergraduate in University of Warwick and Master in Coventry University in 2013~2014 by started to listen music around the world including Chinese, English, German, Japanese and Korean music. Thus, with backgrounds from computing, mathematical and music, this project was successfully planned, scheduled and submitted to Coventry Univesity. Especially, the main concept and the motivation of this project was from two 'similar' K-pop groups: BTS and GOT7.

First of all, I would like to express my sincere gratitude to my Director of Supervisor, Dr. Xiang Fei, and my two top-class supervisors, Dr. Ming Yang and Prof. Kuo-Ming Chao, for all their supports and suggestions to my idea. I would also like to thank Dr. Chaobo He, from South China Normal University (Guangzhou, China), who give me some funding when I'm attending any international conference in China. Next, thanks to all my family members, especially my cousin, David Yee-Fan Zuo (Imperial College London, UK), who helped me proof reading the whole thesis.

Moreover, I would also like to extend my thanks to all the staff, collegues, friends and students who either have shown their interest or have been supported, more or less, to my project. Some people were specially listed below, in 'Surname' alphabetic order:

*Jung Choi* (KAIST, South Korea); *Miguel Francisco* (Rubrik, USA); Dr. *Olivier Haas* (Coventry University, UK); *Yao Li* (Coventry University, UK); Prof. *David Mond* (University of Warwick, UK); Dr. *John Moody* ((ex-) University of Warwick, UK); Prof. *Juhan Nam* (KAIST, South Korea); Dr. *Dianabasi Nkantah* (Coventry University, UK); Dr. *Vasile Palade* (Coventry University, UK); *Saebyul Park* (KAIST, South Korea); Dr. *Sujan Rajbhandari* (Coventry University, UK); *Zhouhemu Tang* ((ex-)Coventry University, UK); *Weihan Xue* ((ex-)Coventry University, UK); *Hongyu You* (Coventry University, UK).

Finally, thanks to all again. It was a brilliant PhD journey with excellent outputs, and I'm truly glad to share the success with everyone.

Best wishes to all,
Hanchao Li

# List of Publications

## Conference Papers

Towards A Hybrid Deep-Learning method for Music Classification and Similarity Measurement – **Hanchao Li**, Xiang Fei, Kuo-Ming Chao, Ming Yang, Chaobo He

(Conference: ICEBE 2016)

A survey of Audio MIR systems, Symbolic MIR Systems and a MDL Demo-System – **Hanchao Li**, Zhouhemu Tang, Xiang Fei, Kuo-Ming Chao, Ming Yang, Chaobo He

(Conference: ICEBE 2017)

MDL & MML: A Coding Scheme for Music Representation and Storage – **Hanchao Li**, David Yee Fan Zuo, Xiang Fei, Kuo-Ming Chao, Ming Yang, Chaobo He Chen

(Conference: ICEBE 2017)

Automatic Note Recognition and Generation of MDL and MML using FFT – **Hanchao Li**, Hongyu You, Xiang Fei, Ming Yang, Kuo-Ming Chao Chaobo He

(Conference: ICEBE 2018)

Theoretical Aspects of Music Definition Language and Music Manipulation Language – **Hanchao Li**, Xiang Fei, Ming Yang,  Kuo-Ming Chao, Chaobo He

(Conference: ICEBE 2019)

## Extended Abstract

Music Data Representation and Information Retrieval Using Vectorbased Similarity Scores – **Hanchao Li**

(Conference: ISMIR 2017)

Music Data Representation Using MDL/MML and Note Recognition Using FFT – **Hanchao Li**

(Conference: ISMIR 2018)

## Journal Paper

A Hybrid Ensemble-Learning Method for Music Classification and Similarity Measurement – **Hanchao Li**, Xiang Fei, Kuo-Ming Chao, Ming Yang, Chaobo He

(Journal: pending) (Extension of the first conference paper)

# Table of Contents

## List of Algorithm, Definition, Lemma, and Theory

## List of Figures

## List of Tables

## Nomenclature & Symbols

### Data Science Techniques

| | |
|---|---|
| A.I./AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| DL | Deep Learning |
| DRL | Deep Reinforcement Learning |
| DT | Decision Tree(s) |
| EML | Ensembled (Machine Learning) |
| ML | Machine Learning |
| RL | Reinforcement Learning |
| RNN | Recurrent Neural Networks |
| SOM | Self-Organized Map |
| SQL | Structured Query Language |

### Mathematical Abbreviations, Varibles, Physical Terms and Units

| | |
|---|---|
| $A$ | Amplitude |
| $\forall$ | for all |
| $B$ | Byte |
| $\mathbb{C}$ | Set of Complex Numbers |
| *cts* (function) | continuous (function) |
| $\exists$ | exist |
| *f, freq* | Frequency or pitch (in *Hz*) |
| *Hz* | Hertz |
| *iff* | if and only if |
| *KB* | Kilo Bytes |
| *Kbps* | Kilobits per second |
| *MB* | Mega Bytes |
| *MSD* | Mean Squred Distance (without square root) |
| $\mathbb{N}$ | Set of Natural Numbers (includes 0, whereas $\mathbb{N}^+$ exclude 0) |
| $\mathbb{Q}$ | Set of Rational Numbers |
| QED, ■ | Quod Erat Demonstrandum (end of a mathematical proof) |
| $\mathbb{R}$ | Set of Real Numbers |
| *sec* | Seconds |
| *sgn()* | Signum function |
| *s.t.* | such that |
| $t$ | Time (in *sec*) |
| *w.r.t.* | with respect to |
| $\mathbb{Z}$ | Set of Integer Numbers |

**Music, Computational Music Terminologies**

| | |
|---|---|
| AAC | Advanced Audio Coding |
| AMT | Automatic Music Transcription |
| CBR | Constant bitrate |
| EM | Electronic Music |
| MIDI | Musical Instrument Digital Interface |
| MIR | Music Information Retrieval |
| MP3 | MPEG1/2 Audio Layer-3 |
| MPEG | Moving Pictures Experts Group |
| MTV/MV | Music (Television) Video |
| SPD | Spectral Pitch Display |
| WAV | Waveform Audio File Format |

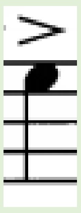
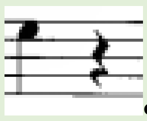
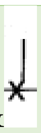
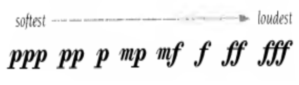
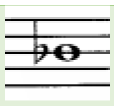
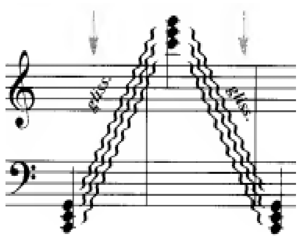
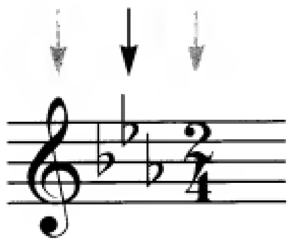
Further music notations and terminologies have shown in Table 0-1.

**Signal Processing Methods**

| | |
|---|---|
| DFT | Discrete Fourier Transformation |
| DSP | Digital Signal Processing |
| FFT | Fast Fourier Transformation |
| FT | Fourier Transformation |
| MFCC | Mel Frequency Cepstral Coefficients |
| SP | Signal Processing |
| STFT | Short Time Fourier Transformation |

**Others**

| | |
|---|---|
| A-MDL | Audio-based Music Definition Language |
| $E^3$MSD | (The architecture of) Expressive, Efficient, and Extendable Music Similarity Detection |
| MDL | Music Definition Language |
| MML | Music Manipulation Language |
| MUCASM | MUsic Classification And Similarity Measurement |
| OMI | Original Music Identifier |
| O-MML | Operational-based Music Manipulation Language |
| S-MDL | Symbolic-based Music Definition Language |
| T-MML | Topological-based Music Manipulation Language |

| **Music Notations and Terminologies** | **Descriptions** | **Music Symbols (if applicable)** |
|---|---|---|
| **Accent** | The note or chord is to be played with slightly more power, compare to unmarked notes. This is one example of an articulation marks. |  |
| **Articulation marks** | This is a group of symbols used to indicate *how* a note or chord is played, alone or in relation to other notes and chords. | n/a |
| **Crotchet / Quarter note (rest)** | A note played for one quarter of the duration of a whole note (semibreve) or rest. |  &  |
| **Dynamics** | Indicates the varying degree of volume or intensity of a note, phrase or section of music. |  |
| **Flat** | Lower a note by one semitone. |  |
| **Glissando** | Indicate the beginning and ending notes of the glissando, can be ascending or descending. |  |
| **Key** | Scale: A-G, sharp or flat, major or minor. (Circle of Fifths) | Details were in Figure 2.5-5. |
| **Key Signature** | A set of sharps or flats, to represent what key the music is in. The correponding note will be raised higher or lower accordingly. |  |

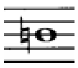
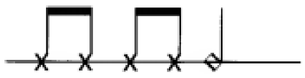
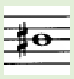
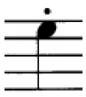
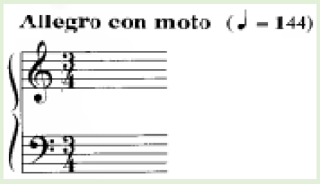
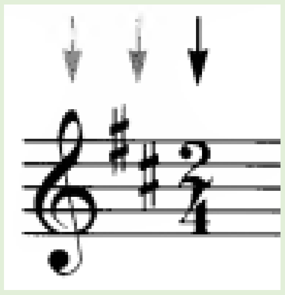
| | | |
|---|---|---|
| **Melody** | Linear tune, voice or line. Series of musical meaningful pitched sounds over the rhythm. | n/a |
| **Natural** | Cancel any flat or sharp previously applied to the note . |  |
| **Pitch** | Perceptual attribute which defines the frequency of the sound wave for the musical melody. | n/a |
| **Rhythm** | Can use an X notehead to represent the non-pitched percussion music part. |  |
| **Sharp** | Raise a note by one semitone. |  |
| **Staccato** | The note or chord is to be played as short as possible. |  |
| **Tempo** | Can be defined by tempo marks. For example, with expression; with a metronome; tempo variations. Determines the rate of speed. |  |
| **Timbre** | Also known as sound 'colour' (tone) related to the recognition of the sound sources. | n/a |
| **Time Signature** | Indicates the meter for a measure(s) or an entire piece, the upper number is the number of beats per measure and the lower number is the duration of the note for one single beat. |  |

*Table 0-1. Musical notations and teminologies (Gerou & Lusk, 1996; Klapuri & Davy, 2006).*

# 1. Introduction

Music, an art form, plays an important role in our daily life. Especially now, as the electronic and the computer technologies have become more powerful, more varieties of music can be generated, not just the classical music with traditional instruments. For example, Electronic Music (EM). On the other hand, as the Internet and the Cloud have become more popular, large repositories of music are available online for public to listen, share, store and download. For example, millions of official Music Videos can be found on YouTube and a lot of CDs can be bought on Amazon. However, due to the large amount of music pieces, people sometimes find it difficult to recognise music from the fragment pieces, or to avoid and detect music plagiarism, as you may find the music that you are listening to is somehow similar to other pieces of music.

Further, from a scientific and philosophic point of view, music is a variety of sounds organized in time with *tonalty* and *aesthetic*, to distinguish between human speech and other non-musical sounds (Kania, 2017). More specifically, *tonalty* involves musical features such as pitch and rhythm whereas *aesthetic* is the interpretation of our feelings, judgement and understanding of music. Thus, there are two major approaches to deal with music. One is using the audio approach and the other one is using the symbolic approach. The audio approach directly represents the musical sound while the symbolic approach captures the musical features. These are the two major music data types which allow computers to store them and Music Information Retrieval (MIR) systems to process them.

Therefore, music data analysis is a complicated but important and interesting topic to investigate.

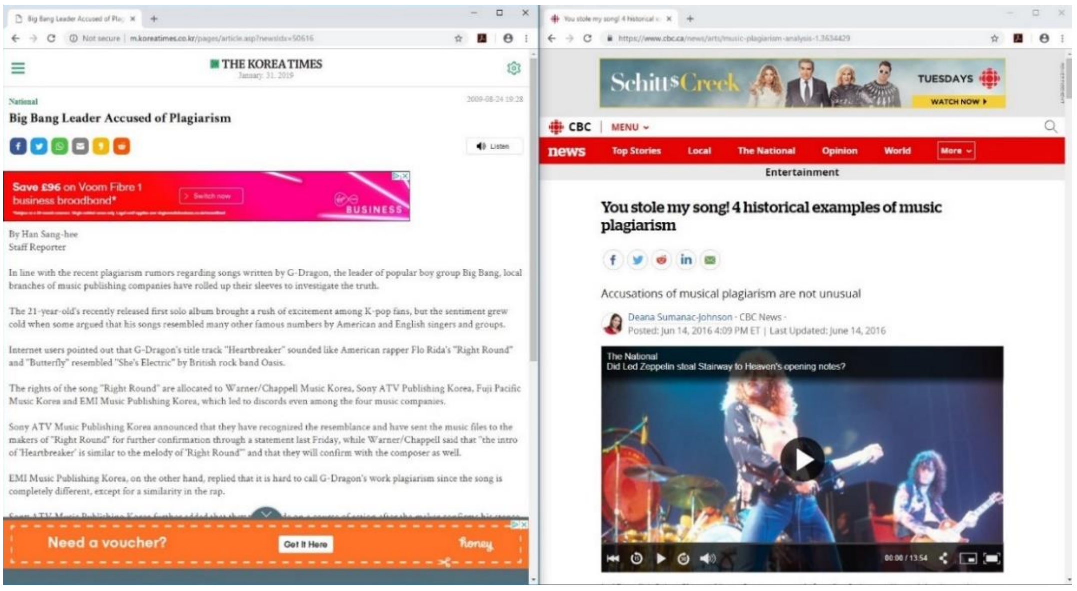
## 1.1. Research Backgrounds

From the music storage and playback's perspective, we mostly store the music in WAV, MP3, AAC format with different bit rate (Brandenburg, 1999) that can affect the quality of the music sound during playback and their file size. These kinds of formats have been classified as audio-based formats as they directly store their audio frequencies over time. Combination of the continuous frequencies with amplitudes can provide more music types and audio signals (Mesaros & Virtanen, 2008). Hence, human vocals are able to be covered. However, we sometimes have encounter problems when faced with these kinds of music formats. For example, when we playback an audio file, to speed it up or down, the soundwaves will be squashed or loosened, leading to the pitch getting higher or lower accordingly. This problem occurs when we need to unify the tempo while generating a remix with two pieces of music which have slightly different tempos.

Alternatively, we can store the music in Musical Instrument Digital Interface (MIDI) format (Rowe, 2009), which has been classified as a symbolic-based format. MIDI can be written from the music version of XML, named MusicXML, which codes the music sheet in the ways of programming code (Good, 2001). Most of the time, music sheets give a better representation of the music in order to allow musicians to compose and read them. However, there are several limits when applying either MusicXML or MIDI to modern Electronic Music. For example, the continuous frequency variation is hard to translate perfectly onto music sheets. When MIDI acts as a symbolic music storage media, it can correspond to Music sheets to reduce the file size compared to any audio format files (Yu, et al., 2002). However, this can only work well for classical music such as piano pieces, instead of modern pop music  (Hrušková & Hvolka, 2011).

From the music application's perspective, there are two scenarios, music search and music plagiarism.

For the music search scenarios, there are three types of music search engines. Tag-based, or meta-data-based MIR systems provide search services by various categories, such as by title, by artist, by album etc., even Google, non-MIR systems sometimes can do the job for us. These tag-based search engines are effective, however they are based on an assumption that we have certain information about the music we want to find. This assumption does not always hold, e.g., if we are in a public area, or when we are watch TV shows at home, we may hear some backgound music for the first time. On the other hand, audio content-based music search engines, e.g., Kugou (Hu, 2018) and Shazam (Typke, et al., 2005), allow us to search for music based on the musical content rather than the prior information of meta-data. However, if the music piece we listened to is not the original piece, for example, a re-composite or re-performance version (e.g. live version), these kinds of systems are normally unable to find the original version, due to the uniqueness of the Audio Fingerprints method the systems use (Typke, et al., 2005). In some cases, such as in a disco club, the backgound music is often a remix. A remix is


Big Bang Leader Accused of Pla
m.koreatimes.co.kr/pages/article.asp?newsIdx=50616
Listen
Save £96 on Voom Fibre 1 business broadband*
Switch now
BUSINESS
Need a voucher?
Got It Here
Honey
You stole my song! 4 historical
https://www.cbc.ca/news/arts/music-plagiarism-analysis-1.3634429
Schitt$Creek
TUESDAYS
WATCH NOW
CBC
MENU
news
Top Stories
Local
The National
Opinion
World
More
Entertainment

THE KOREA TIMES
January 31, 2019

National

2009-08-24 19:28

Big Bang Leader Accused of Plagiarism

By Han Sang-hee
Staff Reporter

In line with the recent plagiarism rumors regarding songs written by G-Dragon, the leader of popular boy group Big Bang, local branches of music publishing companies have rolled up their sleeves to investigate the truth.

The 21-year-old's recently released first solo album brought a rush of excitement among K-pop fans, but the sentiment grew cold when some argued that his songs resembled many other famous numbers by American and English singers and groups.

Internet users pointed out that G-Dragon's title track "Heartbreaker" sounded like American rapper Flo Rida's "Right Round" and "Butterfly" resembled "She's Electric" by British rock band Oasis.

The rights of the song "Right Round" are allocated to Warner/Chappell Music Korea, Sony ATV Publishing Korea, Fuji Pacific Music Korea and EMI Music Publishing Korea, which led to discords even among the four music companies.

Sony ATV Music Publishing Korea announced that they have recognized the resemblance and have sent the music files to the makers of "Right Round" for further confirmation through a statement last Friday, while Warner/Chappell said that "the intro of 'Heartbreaker' is similar to the melody of 'Right Round'" and that they will confirm with the composer as well.

EMI Music Publishing Korea, on the other hand, replied that it is hard to call G-Dragon's work plagiarism since the song is completely different, except for a similarity in the rap.

You stole my song! 4 historical examples of music plagiarism

Accusations of musical plagiarism are not unusual

Deana Sumanac-Johnson · CBC News ·
Posted: Jun 14, 2016 4:09 PM ET | Last Updated: June 14, 2016

The National
Did Led Zeppelin steal Stairway to Heaven's opening notes?

*Figure 1.1-1. Two news reports about music plagiarisms (Han, 2009; Sumanac-Johnson, 2016).*

a combination of different music pieces with alternations, e.g., adding notes, removing notes or modifing melodies will make it harder to recognise the origin using an audio content-based music search. If you already know the symbolic melody for it, then symbolic content-based MIR systems such as Musipedia (Typke, et al., 2005) can allow you to search for the music pieces, even the variations, but majority of available symbolic-based MIR systems are only for classical music. Therefore, there is more room for the existing content-based music search engines to improve, so the searching result or recommendations can be based on both audio and symbolic music contents, in contrast with the tag-based music seach engines (Song, et al., 2012; Knees & Schedl, 2013).

For the music plagiarism scenario, there are a lot of worldwide news reports about music plagiarism. Figure 1.1-1 shows two worldwide news reports (Han, 2009; Sumanac-Johnson, 2016) as examples. From Han's report (Han, 2009), G-Dragon, Ji-yong Kwon as the original name, has released a new title track "Heartbreaker" [33]* from the album 《Heartbreaker》 in 2009. This track has been pointed out to have similarities to Flo Rida's "Right Round" [34] by netizens. After communication between the artists (and the companies/representatives), G-Dragon featured Flo Rida in his new version of the song titled "Heartbreaker" [35] in his 2010 album 《Shine a Life》 (Gaon Chart (2010.03.28-2010.04.03), 2010). Similar to another track of his, "Butterfly" [36] under the same album, this song track was reported to have similarities to Oasis's "She's Electric" [37]. This time,

* music track references in squre bracket is on page 111.

however, the incident has been cited as a plagiarism example in high school textbooks despite all charges have been lifted against him (Esther, 2013). From Deana's reports (Sumanac-Johnson, 2016), for one of the four historical songs, 'Stay with me' [38], was accused of music plagiarism in 2015, Sam Smith claimed that he has not heard Tom Petty's song, "I Won't Back Down" [39] which was published before he was born. As there is a large increase in the amount of music around the world, it is difficult for people to listen to all the existent music before the release of their albums, to avoid plagiarism claims. Similar situations have been applied to the academic papers, e.g., conference papers, journal papers, and thesis. However, for academic papers, we have a system called '*Turn-it-in*', to identify and highlight the word patterns with the percentage of similarity scores as output, such that we can guide us to judge how the newly submitted paper is similar to the existing papers. Thus, there is a need to build a similar plagiarism detection system to the music area (Park, et al., 2005; Dittmar, et al., 2012; Shin, et al., 2016). We need to find a way to code the songs such that we can highlight the similar patterns based on the similarity scores between the codes. Every time a new song enters the music database, by using this approach, we can identify and highlight any similar musical patterns shared by other original musical tracks. Hence, the composer can pre-reference or be notified before being accused by other composers, and receive the copyright fee with confidence. In order to build this system, we need to analyse the core musical content.

## 1.2. Research Questions

From the research background in Section 1.1, in order to improve the performance of the existing MIR system or to implement a tool for music plagiarism idenfitication, there is one fundamental research question which needs to be answered:

**"How to improve the performance of detecting similar music pieces?"**

This can be further broken down into two questions, one of them concentrates on the music coding scheme, and the other one focuses on the similarity evaluation:

1. "*How to precisely and expressively code the music in such a way that we can keep the audio feature and reduce the file size?*"
2. "*How to effectively measure the similarities in such a way that we can identify the original music from its variations?*"

## 1.3. Research Aims and Objectives

From the research questions stated in Section 1.2, below list the aims and objectives for this project:

- Provide a better understanding for the musical content.
    - Evaluate the features and limitations for the audio (e.g. MP3) and symbolic (e.g. MIDI) approaches/formats.
    - Design a structured and expressive music data model that can be used to extract the musical features, e.g. melody, rhythm.
    - Develope prototypes for proof-of-concept.
- Be able to identify original music tracks from the variations.
    - Design an algorithm which can evalute the similarities between two pieces of music pieces. This can be between two pieces of original music, or between one variation and one piece of original music. Based on which, we apply the algorithm to find the original tracks from the variations.
    - Develope prototypes which demonstrate the core functionalities including benchmarking.

## 1.4. Novelty, Contribution and Challenge

As described in the Section 1.1 - 1.3, our principal contribution is to retain the advantages from both audio and symbolic sides while address individual shortcomings, e.g., a high fidelity music but with a smaller file size. This is achieved by outlining a new architecture, named $E^3$MSD (Expressive, Efficient, and Extendable Music Similarity Detection). This architecture consists of two music modelling languages, and one algorithm using the Ensembled Machine Learning (EML) method called MUCASM (MUsic Classification And Similarity Measurement). There are several prototypes that have been built which are used to demonstrate the three features of the proposed architecture.

To the best of our knowledge, this is the first attempt to construct a general music data model, which is interdisciplinary between the three subject areas (Computer Science/Data Science, Mathematics, Music) and two systems (Audio and Symnolic), and further to apply modern machine learning techniques for the computers to understand the music in the similar ways human beings do.

## 1.5. Thesis Structure

The rest of the thesis is structured as the following:

- Chapter 2 summarizes the features and limitations for the existing technologies in the music area. This includes case studies and general literature reviews. It also provides a short overview for all the important techniques and terms in Computer Science, Mathematics and Music. These fundamental theories behind the techniques and terms will not be further described or explained in later chapters.
- Chapter 3 outlines the methodology for the project and the main work flow for the architecture E$^3$MSD.
- Chapter 4 introduces the structured and extensible music data model, named Music Definition Language (MDL) and Music Manipulation Language (MML). This includes the example of MDL&MML, and any theoretical proofs; followed by demonstrations on how we can playback the MDL and MML files and automatically convert the audio into the MDL and MML, using Signal Processing techniques.
- Chapter 5 covers the design, implementaiton and test of MUCASM for Original Music Identifier (OMI), where we classify music variations back to their origins using the Reinforcement Learning based EML method.
- Chapter 6 further evaluates MUCASM and benchmarking.
- Chapter 7 gives the final conclusion and provides general future works.

At the end of the report, there is a ‘Music Copyright and References’ section. This section includes two lists of all the music/academic papers which have been mentioned/used in the report. Finally, all other relevant materials, including the project management, are inside the ‘Appendix’ section.

# 2. Literature Review

In this chapter, we investigate existing MIR systems, include musical analysis & storage methods and other recent modern techniques which have been applied in the music related area, followed by a section of background knowledge overview.

## 2.1. Music Information Retrieval Systems for Search Engines

Music Information Retrieval (MIR) systems discover useful information from music pieces. Hence, the tasks for an MIR system can be pattern recognition, genre detection, mood evaluation, instrumentation, music recommendation, plagiarism detection tools based on similarity and music search engines (Casey, et al., 2008; Lamere, 2008). In other words, MIR can search unknown songs, maintain copyright and find similar style music pieces.

For example, if 'A' is an original song, 'B' is a re-performance of 'A', 'C' is a re-composite of 'A', 'D' is another original song which is completely different to 'A', 'E' is an original song that is somehow similar to 'A' and has not been categorized as plagiarism, 'F' is a remix of 'A' and 'E'. Then ideally, the following should happen for a music search engine or a plagiarism detection tool:

If we input 'B' or 'C', 'A' should be our second output with a relatively high similarity score, as re-performance is a simple variation such that the main melody flow remains largely unchanged, whereas re-composite does vary the main melody flow although the main pattern is still recognizable.

The overall similarity score between 'A' and 'D' should be much lower than 'A' and 'E'. Moreover, 'A' and 'E' may have certain high similarity scores, e.g., rhythmic similarity. By having this property, we can make the remix song 'F' from 'A' and 'E'. Thus, there are lots of mashup remix on Youtube. For example, Miguel Francisco (Youtube account name: Miggy Smallz, refer to acknowledgement) has a number of K-Pop mashup remix tracks (Francisco, 2013-2019), such as a combination of GOT7's "Never Ever" [40] with BTS's "Not Today" [41] and KARD's "Don't Recall" [42] (Francisco, 2017). The more features shared between two or more pieces of original songs, the smoother the mashup remix. Another good example of a mashup remix would be Green Day's "Boulevard of

Broken Dreams” [43] with Oasis’s “Wonderwall” [28] (OasisIndie94, 2011). Thus, for an MIR system, both ‘A’ and ‘E’ should be the output if we are requesting song ‘F’.

However, MIR systems can be tag-based or (music) content-based. A social-tag-based music recommendation system is an example of a tag-based MIR system, where the tags can be an artist, album or a song (Lamere, 2008). For (music) content-based MIR systems, there were several content-based MIR systems that exist, typical examples being Shazam and Musipedia. These two MIR systems use different input types with different methods. As previously mentioned, these can be categorized into the following two major categories: audio-based and symbolic-based (Typke, et al., 2005; Casey, et al., 2008). Audio-based systems primarily use a technique called Audio Fingerprint (Typke, et al., 2005), while symbolic-based systems use several methods such as contour-based, melody-based and rhythm-based (Typke, et al., 2005) with music notation and symbolic representation (Casey, et al., 2008).

Therefore, we start with reviewing the audio and symbolic approaches individually, which includes the traditional literature review and a case study based system investigation. This is followed by comparing these two approaches and outline their features and limitations.

### 2.1.1. Audio

For audio-based storage, the common file extension would be “.wav”, “.mp3” and “.aac”. For example, online songs (music with lyrics) can be downloaded and stored using those file formats. Based on these type of files, audio fingerprinting is an ideal approach as it can be deterministically generated from an audio signal. Hence, the key technique used in the system is the building of an audio fingerprint catalogue. Each music piece corresponds to a unique fingerprint by sampling the audio or digitized signals, which is used to identify the song by matching the fingerprint (Haitsma & Kalker, 2002; Haitsma & Kalker, 2003; Cano, et al., 2005; Duong & Duong, 2015). Shazam is one of the typical audio-based search engines for music recognition applications (Shazam Homepage, 2019), whereas Kugou is a chinese audio-based music player (Kugou Homepage, 2019), but also has the funcionality of searching the song by listening to the music, naming ‘tinggeshiqu (听歌识曲)’, as shown in Figure 2.1-1 (2.1-1a & 2.1-1b). Thus, we start by investigating these audio-based MIR systems (including Kugou, Shazam),

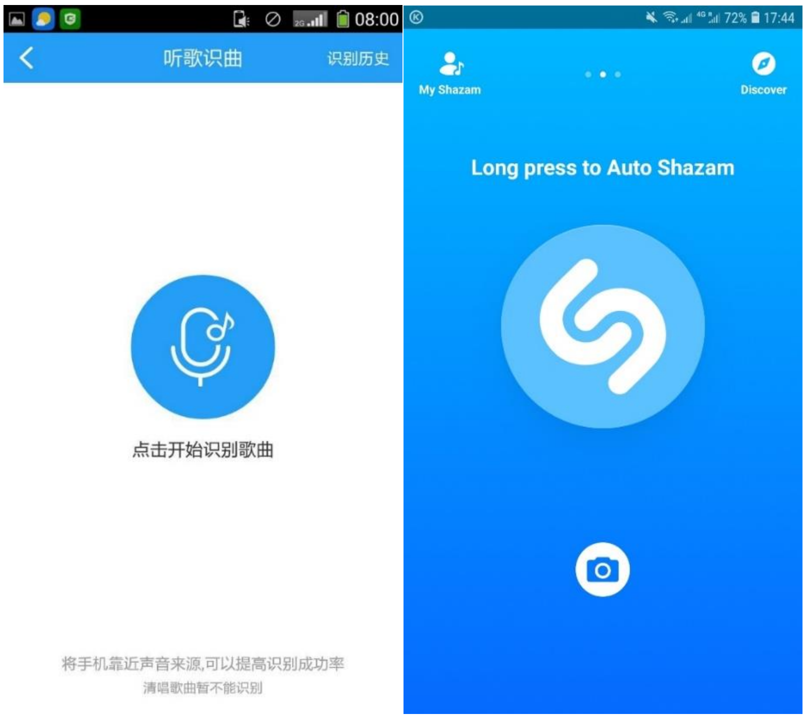


*Figure 2.1-1a. Kugou* *Figure 2.1-1b. Shazam*

*Figure 2.1-1. Home page for audio-based MIR systems: Kugou and Shazam (Android Applications Screenshots).*

followed by analysing the features and limitations from several papers, and ending with a summarization derived from both case studies and literature review.

#### 2.1.1.1. Case Studies

We have tested music tracks [1] to [12] from the music copyright list (page: 113) over the following systems: Baidu Music (BM) (Baidu Music Homepage & Qianqian Music Homepage, 2019), Kugou Music (KG) (Kugou Homepage, 2019), Kuwo Music (KW) (Kuwo Homepage, 2019), Music Radar (MR) (Musicradar Homepage, 2019), Netease Cloud Music (NC) (Netease Cloud Music Homepage, 2019), QQ Music (QM) (QQ Music Homepage, 2019), Shazam (SZ) (Shazam Homepage, 2019), SoundHound (SH) (SoundHound Homepage, 2019), and Track ID (TI) (TrackID Homepage, 2017). We play the music from a random starting point, and test whether the system can recognize the music or not. All those twelve songs have lyrics. The result is shown in Table 2.1-1.

Note: "✔" represents correct matching; "Ø" represents no result as it is not in the database; "✕" represents wrong matching and "A" stand for the overall accuracy. As we have tested 'TrackID' before it closed at 15 September 2017, we grayed out our last row.

| Name | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] | [12] | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BM | Ø | Ø | ✓ | ✓ | Ø | Ø | Ø | Ø | Ø | Ø | ✓ | ✓ | 33 |
| KG | Ø | ✓ | ✓ | ✓ | ✓ | ✓ | ✕ | Ø | Ø | Ø | ✓ | ✓ | 58 |
| KW | Ø | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✕ | ✓ | ✓ | ✓ | ✓ | 83 |
| MR | Ø | ✓ | ✓ | ✓ | ✓ | Ø | Ø | ✓ | ✓ | Ø | ✓ | ✓ | 67 |
| NC | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 100 |
| QM | ✕ | ✓ | Ø | ✓ | Ø | ✓ | ✓ | Ø | ✓ | ✓ | Ø | ✓ | 58 |
| SZ | ✓ | ✕ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | Ø | ✓ | ✓ | 83 |
| SH | ✓ | Ø | ✓ | ✓ | Ø | ✓ | ✓ | Ø | Ø | Ø | ✓ | ✓ | 58 |
| TI | Ø | Ø | ✓ | ✓ | Ø | Ø | ✓ | ✓ | Ø | ✓ | ✓ | ✓ | 58 |

*Table 2.1-1. Recognition result for audio-based MIR systems ([1]-[12]) (Tang, 2016).*

From Table 2.1-1, we can immediately conclude that the Netease Cloud has the highest music recognition accuracy of 100%, and only four wrong matchings exist in the table. However, despite that there is no white box testing with those applications, the following findings can be concluded with some further black box tests:

Firstly, all the systems can only find the songs if and only if the music track exists in the database. This indicates that the audio-based music recognition applications have high precision, whereas precision is the fraction of correctly recognitions over all cognitions. Some systems have the a precision of 100% and others have 87.5% or 90.9%.

Secondly, for the same songs with variations, e.g., different singers, different musical instruments, different tempo or contain background noises, it will be difficult for the audio-based MIR systems to recognise them. For example, if another singer covers one singer's original song on a live TV show, then it is hard to find the original singer's version under the audio-based MIR systems, as the audio fingerprints are different.

Finally, all applications in this case study only support up to a maximum of 20 seconds. This indicates the input length is another factor that affects the matching accuracy. For example, if two songs share a similar audio fingerprint for partial melodies, then the system may output a mismatched original song when inputting the shared melodies. Moreover, with the same input length, the more distinctive features the music has, the higher the likelihood of the original music to be correctly recognized.

#### 2.1.1.2. *Literature Critiques*

In this section, we have chosen five papers to write a short critique about. These papers were collected so that some of them are highly cited but older papers while others are more recent ones that are less cited. Here are short critiques for those five papers, including features and limitations of the audio-based method claimed by those papers:

Casey el al. (2008) have introduced that the current retrieval systems can handle millions of music tracks. However, the systems need to aim at even larger online music collections. It outlines the problems of content-based MIR, and explores the state-of-the-art methods using audio cues (e.g., query by humming) and other cues (e.g., music notation and symbolic representation). Moreoever, it identifies some of the major challenges in MIR systems. However, this paper does not provide the resolving methods.

Typke el al. (2005) summarized all the existing content-based MIR systems, i.e., several music search engines. This paper includes three methods for searching symbolic data and four methods for searching the audio data from seventeen existing MIR systems. Furthermore, it contains a summary table to show the type of input (audio/ symbolic), the type (audio/ symbolic) and level of matching (exact/ approximate/ polyphonic), and certain features each MIR system involves. We have quoted the summary table and shown them in Table 2.1-2 and highlighted MIR systems in blue for audio-based, red for symbolic-based, green for audio input with symbolic matching, and yellow when both inputs are accepted.

From Table 2.1-2, only three MIR systems allow the input to be both audio and symbolic. However, all of these three MIR systems can only apply to the monophonic approximate matching. As a result of this, it is difficult for the system to search the exact polyphonic music tracks.

One of the systems among the rest of the fourteen MIR systems can do polyphonic, exact and approximate matching and is provided with nine algorithms. However, this MIR system, C-Brahams, is not an audio-based MIR system. As a drawback, people who do not know the symbolic version of the rhythm cannot find the exact songs. Hence, there is a need to represent the music in a such a way that we can extract features from the audio files directly with various matching methods.

| **Name** | **Input** | | **Matching** | | | | | **Features** | | | | | | | | **Indexing** | **Collection size (Records)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Audio | Symbolic | Audio | Symbolic | Exact | Approximate | Polyphonic | Audio-Figerprints | Pitch | Note Duration | Timbre | Rhythm | Contour | Intervals | Other | | |
| **Audentify!** | • | | • | | | • | • | • | | | | | | | | Inverted files | 15,000 |
| **C-Brahms** | | • | | • | • | • | • | | • | • | | • | | • | | None | 278 |
| **CubyHum** | • | | | • | | • | | | | | | | | • | | LET | 510 |
| **Cuidado** | • | | • | | | • | • | | | | • | • | | | • | Not descriped | >100,000 |
| **GUIDO/ MIR** | | • | | • | | • | | | • | • | | • | | • | • | Tree of transition matrices | 150 |
| **Meldex/ Greenstone** | • | • | | • | | • | | | | | | | • | • | | None | 9,354 |
| **Musipedia** | • | • | | • | | • | | | | | | | • | | | Vantage objects | >30,000 |
| **notify! Whistle** | • | • | | • | | • | | | • | | | • | | | | Inverted files | 2,000 |
| **Orpheus** | | • | | • | | • | • | | • | • | | • | | • | | Vantage objects | 476,000 |
| **Probabilistic "NameThatSong"** | | • | | • | | • | | | | | | | | • | • | Clustering | 100 |
| **PROMS** | | • | | • | • | • | | | • | | | • | | | | Inverted files | 12,000 |
| **Cornell's "QBH"** | • | | | • | | • | | | | | | | • | | | None | 183 |
| **Shazam** | • | | • | | • | | • | • | | | | | | | | Fingerprints are indexed | $>2.5 * 10^6$ |
| **SOMeJB** | • | | • | | | • | • | | | | | | | | • | Tree | 359 |
| **SoundCompass** | • | | | • | | • | | | | • | | • | | | | Yes | 11,132 |
| **Super MBox** | • | | | • | | • | | | | • | | • | | | | Hierarchical Filtering | 12,000 |
| **Themefinder** | | • | | • | • | | | | | • | | | • | • | | None | 35,000 |

*Table 2.1-2. Content-based MIR systems (Typke, et al., 2005).*

Furthermore, this paper (Typke, et al., 2005) has introduced one typical example of audio-based MIR systems, naming Shazam. Shazam uses an audio fingerprint method to find the matching. This method generates reproducible landmarks and extracts features which describe the short segments of recordings as fingerprint tags which also characterize its location. However, as the landmarks and tags can be affected by low quality speakers, microphones or background noise, thus, the accuracy would sometimes not be that high. For some of the other audio-based MIR systems, this paper has mentioned that they use either set-based or string-based methods for finding matchings in polyphonic music from the audio input. One of the features for the set-based method is that the music is viewed as an unordered set of events where each has its own properties described. However, an unordered set will lead to a low accuracy search, as music is a sound sequence over time and a set loses information from a sequence. On the other hand, the string-based method treats musical notes as a string-based sequence, but only for monophonic melodies.

Thus, Sri Ranjani, et al. (2015) have tested audio-based MIR system: Shazam, with Indian songs, and obtained a retrieval accuracy of 85% in their paper.

Duong & Duong (2015) have provided the architecture for the existing audio fingerprinting method. The whole architecture consists of a feature extraction stage, feature modeling stage and feature matching stage, as shown in Figure 2.1-2. For the feature extraction stage, the paper introduced several audio-based features using various methods from signal processing, whereas for the feature modeling stage, the paper introduced several statistical models. Furthermore, the paper claimed that by combining several features will benefit the system: obtaining a robust and compact audio signature. On the other hand, the paper mentioned that, to use those statistical models, will reduce the global statistical redundancy of feature vectors, which decreases the size of the fingerprint. However, this paper does not comment on whether the reduction of fingerprint size will affect the matching accuracy or not, i.e., the searching result for an audio-based music search engine, as this paper concentrates on the audio fingerprint design only.

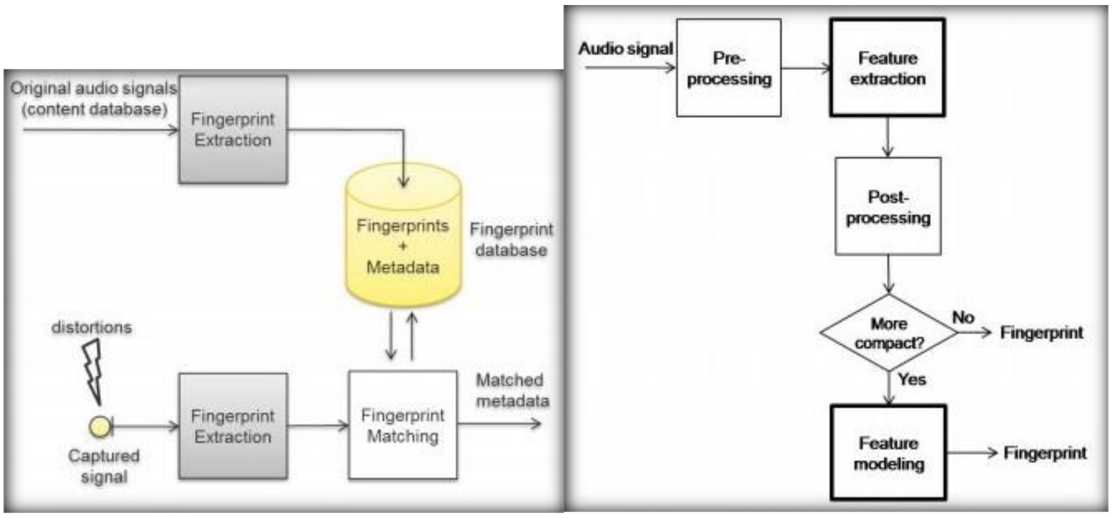


*Figure 2.1-2. Architecutre of audio fingerprinting system with the fingerprint design (Duong & Duong, 2015).*

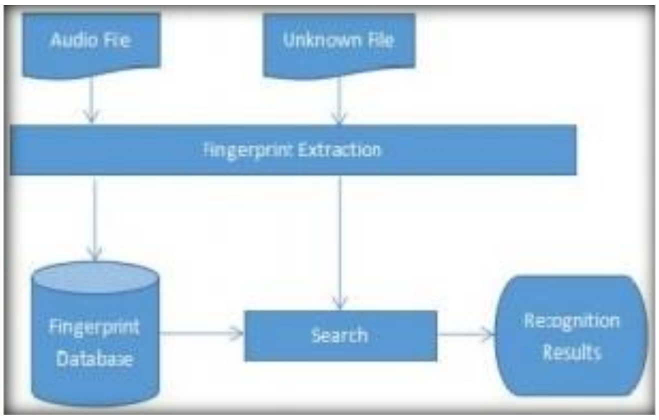


*Figure 2.1-3. Work flow for an audio-based music recognition system (Fan & Feng, 2016).*

Finally, Fan & Feng (2016) have actually construct the work flow for an audio-based music identification/recognition system, as shown in Figure 2.1-3. Furthermore, they described the steps of fingerprint generation in detail: start with FFT transformation; followed by peak extraction from frequency spectrogram; next, extract the time difference, the starting frequency and the "fingerprint" data; and finally, ends with fingerprint generation using hash algorithm. After comparing with the fingerprint database by computing the similarity scores between two fingerprints, it outputs the songs with the highest similarity scores. From their experiment and analysis, their music identification system has strong robustness, fast identification with high recognition accuracy of 45 songs. However, they have pointed out that the longer the audio, the longer the time taken for recognition.

From those case studies and critiques, here are some critical points about the main feature and the major limitation for audio files and audio-based MIR systems:

- ✓ MIR systems can carry out various tasks involving genre, lyrics, mood, and timbre.
- ✓ For audio-based music search engines, each song has a unique fingerprint, similar to the fingerprints of human beings. Thus, the music track can be correctly recognized if the input song's fingerprint matches with the one in the existing MIR database.
- – The approach is less tolerant to background noises and music variations. Re-composite and re-performance, for example, were have difficulty in finding their original music pieces for an audio-based MIR systems, even if the original piece has been stored in the music database.
- – The longer the audio length, the larger the fingerprint, the longer processing time is required for recognition.

### 2.1.2. Symbolic

For the symbolic-based storage, the common file extension would be ".midi", where, the typical symbolic-based music melody search engine would be Musipedia (Typke, 2002). Symbolic files, like MIDI, is different to an audio file in either analogue (e.g. WAV) or digital (e.g. MP3) form, as we are using only the protocols to capture the details of a musical note from the discrete set of the music signals rather than the signals themselves. Hence, symbolic files can easily be stored and manipulated compared to any audio files as storing those musical events and parameters only take less space (McKay & Fujinaga, 2006; Good & Mills, 2015). However, symbolic-based MIR systems only suitable for music-sheet-convertible music tracks (Saxena, et al., 2018). Hence, the key step of symbolic-based approaches is how to code music using various symbols (Loy, 1985; Lubiw & Tanur, 2004; Valero & Quereda, 2010; Walder, 2016). Based on the type of symbolization, various content-based similarity scores can be evaluated, e.g., based on the contour, the melody, the rhythm, etc. (Typke, et al., 2005). Thus, we start by investigating the cases of these symbolic-based MIR systems (including Musipedia), followed by analysing the features and limitations from several papers, and ending up with a summarization.

#### 2.1.2.1. Case Studies

We have tested music tracks [13] to [23] from the music copyright list (page: 113) over the following systems: Musipedia (M) (Typke, 2002), Best Classical Tunes (B) (Gomersall & Clarke, n.d.), and Themefinder (T) (Huron, et al., n.d.). The result is shown in Table 2.1-3.

| Name | Type | [13] | [14] | [15] | [16] | [17] | [18] | [19] | [20] | [21] | [22] | [23] | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M | M | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 100 |
| | C | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 100 |
| | R | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | 27 |
| B | T | ✓ | ✓ | Ø | Ø | Ø | ✓ | Ø | ✓ | Ø | ✓ | ✓ | 55 |
| T | P | ✓ | ✓ | ✓ | Ø | Ø | ✓ | ✓ | Ø | Ø | ✓ | Ø | 55 |
| | S | ✓ | ✓ | ✓ | Ø | Ø | ✓ | ✓ | Ø | Ø | ✓ | Ø | 55 |
| | GC | ✓ | ✓ | ✓ | Ø | Ø | ✓ | ✓ | Ø | Ø | ✓ | Ø | 55 |
| | RC | ✓ | ✓ | ✓ | Ø | Ø | ✓ | ✓ | Ø | Ø | ✓ | Ø | 55 |

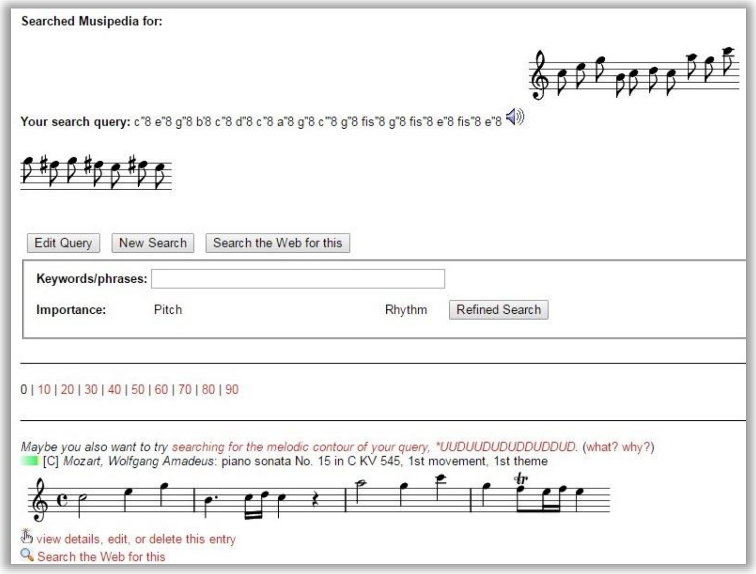
*Table 2.1-3. Recognition result for symbolic-based MIR systems ([13]-[23]).*

Note: " ✓ " represents correct matching; "Ø" represents no result as it is not in the database; "✗" represents wrong matching and "A" stands for the overall accuracy. For the 'type' column, 'M' stands for the Melody search; 'C' for the Contour search; 'R' for the Rhythm search; 'T' for the Theme search; 'P' for the Pitch search; 'S' for the Scale search; 'GC' for the Gross Contour search; and 'RC' for the Refined Contour search.

Here are the detailed evaluations for each symbolic-based music search engine, including their features and limitations.

For 'Musipedia', as shown in Table 2.1-1 and Table 2.1-3, it covers the melody search, the contour search and the rhythm search. Contour search is where we input whether the note is going up, down or remains the same with respect to the previous note. Rhythm search is where we exclude the pitch information from the melody.

When carrying out the melody search, independent with the note duration and key signature, the symbolic-based MIR system can output the melody in the key that the



Searched Musipedia for:

Your search query: c"8 e"8 g"8 b'8 c"8 d"8 c"8 a"8 g"8 c"'8 g"8 fis"8 g"8 fis"8 e"8 fis"8 e"8

Keywords/phrases:

Importance: Pitch Rhythm

0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90

*Maybe you also want to try searching for the melodic contour of your query, *UUDUUDUDUDDUDDUD.* (what? why?)

🟩 [C] *Mozart, Wolfgang Amadeus*: piano sonata No. 15 in C KV 545, 1st movement, 1st theme

*Figure 2.1-4. Melody search in Musipedia for melody containing turning note [18].*

original melody of the music track has stored in the Musipedia database with corrected note durations, as we had input all the melodies in the C key and all the notes were quavers (8th note). Furthermore, if the original melody contains notes with certain articulation marks, e.g., turning notes, Musipedia would be able to find the corrected melody independent to the input expansion of those music notations, as shown in Figure 2.1-4. Thus, like Table 2.1-2 summarized, the matching type for Musipedia is approximate. However, if the original melody contains multiple notes, e.g., chords, the search engine can find the corrected melody occasionally, similar results were obtained when carrying out the corresponding contour search.

Lastly, for the rhythm search, low rhythm retrieval rate, three out of eleven, have been obtained. This suggests that for melodies, pitch information is much more important compared to the rhythm information.

For 'Best Classical Tunes', it only covers theme search only, which corresponds to the melody search in Musipedia. The theme input can be scale-free, i.e., all in C major or A minor, set-free, i.e., narrowed down to twelve different symbols, and duration-free. Moreover, this MIR system is capable for expansion of the turning notes. Thus, like Musipedia, this symbolic-based MIR system is using approximate matching. However, Best Classical Tunes obtained low accuracy rates compared to Musipedia. This is due to the size of the music database, so if we look at both systems' precision, they were both 100%.

As for ‘Themefinder’, it provides pitch search (similar to the melody search in Musipedia, but without the duration information), interval search (detailed contour search), scale search (similar to the pitch search, but using the modulo 7 system instead of 12), gross contour search (same as the contour search in Musipedia) and refined contour search (extension of the gross contour search). It also provided options that the input is from the beginning of the theme or anywhere inside the theme. For all these searches, up to 100 songs are shown in the result list.

Firstly, for the pitch searches, the system is very sensitive to the key notation. For example, despite F sharp and G flat sharing the same piano key, for a song in G key, G flat would not output the original melody. This system, unlike Musipedia, does not allow the expansion of the turning note. Furthermore, it is scale sensitive. For example, A minor is different to C major, despite the fact that they were related. Therefore, Themefinder provides exact matching only, the same conclusion that is shown in Table 2.1-2.

Secondly, interval search does not appear to work as we have input the example track showed on the website. In fact, this search required users to have strong music knowledge.

Lastly, the difference between gross contour (GC) and refined contour (RC) were melodies with different RC that may have they same GC. For example, the first two bars of track [13] and track [15] having the same GC sub-sequence, ‘uuuuudud’, but track [13] has the following RC sub-sequence ‘uuuuuDUd’, whereas track [15] has the following RC sub-sequence ‘‘uuuuudUd’. In GC, ‘u’ stand for up; ‘d’ for down and ‘s’ for the same. In RC, ‘u’, ‘d’ and ‘s’ stand for up by step, down by step, and remained the same respectively, whereas ‘U’, ‘D’ stand for up by leap, down by leap respectively. Again, as a drawback, these kinds of input require users to know the outline of the music melody and have understood the rules.

Finally, for all these symbolic-based MIR systems, only pre-symbolically-stored melodies can be searched. Thus, we need to carry out the transcription for those music pieces, i.e., extract the melody from the audio files and convert into a sequence of piano-roll-based symbols. Therefore, most of the tracks [13] to [23] were classical piano music pieces, rather than pop songs.

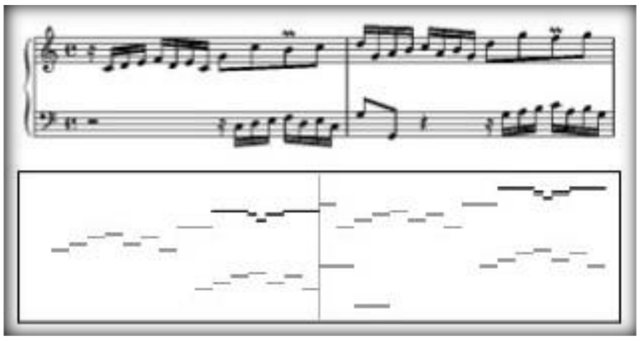


*Figure 2.1-5. Line segment representation for the first two bars of J.S. Bach, Invention 1, C major, BWV 772 [44] (Lubiw & Tanur, 2004).*

#### 2.1.2.2. Literature Critiques

In this section, similar to Section 2.1.1.2, we have chosen five papers and provided short critiques for each of them, where features and limitations of the symbolic-based method claimed by those papers are included.

Based on MIDI representation, Lubiw & Tanur (2004) search for the similar patterns inside the same piano music use line segments and highlights those patterns. The core of this approach is using the mathematical term: Geometric Translation, in such a way that once we have shifted the highlighted patterns up or down along the pitch axis, and stretched or squashed along the time axis, the patterns became perfectly identical. They have provided with an example to illustrate the relationship between the piano sheet and the line segment, as well as bolded two line segments for similar melody patterns. This has shown in Figure 2.1-5.

Despite this approach being able to deal with note with various articulation marks such as dots, ties and syncopations and even polyphonic piano music pieces, this approach is still limited to the discrete pitch set, as the glissando note for the modern (electronic) music, where we consider the continuous pitch set, cannot be easily represented and analyzed using it.

Further from the music representation, they have evaluated the computational time complexity for their pattern matching algorithm to be $O\big(nm(d + \log m)\big)$, where d is the size of the pitch set, n is the score size and m is the pattern size. Hence, they have claimed that this algorithm works faster only for small patterns, as the factor $O(nm)$ is hard to improve since it is at least as hard as the 3-SUM or Segments Containing Points problems. Further details about those problems, please refer to the paper.

Two years later, McKay & Fujinaga (2006) have evaluated a software package that extract features from the MIDI files, where the extracting language is known as the jSymbolic. It analyses the library containing 160 high-level features which can be used to classify music by evaluating the music similarity scores. These features can be categorized into the following groups: instrumentation; texture; rhythm; dynamics; pitch statistics; melody and chords. As jSymbolic is currently compatible with MIDI files, they have mentioned that their target is to expand its functionality such that it can process other symbolic formats. Later in 2016, they released jSymbolic2 in their newest article (McKay, et al., 2016), which is a significant expansion of the first version of jSymbolic, as features can also be extracted from MEI files and, if using a Rodan conversion workflow, MusicXML.

From Music XML, Hrušková & Hvolka (2011) have introduced a melodic and rhythmic vector format approach in representing and finding similarities between two monophonic music files. This can be passed through a “Melody Comparator” function. Unfortunately, this paper does not provide the details of the experiment for evaluating the similarities between two music pieces, and the vector format they have introduced, has limited information stored for the individual note.

However, Valero & Quereda (2010) have introduced the tree structure to represent the music instead of the vector format. The layers of the tree define the duration length of individual nodes from the melody and the root of the tree defines the value of the individual nodes. An example is shown in Figure 2.1-6, where each note is using the modulo 12 system and the dummy note has been shaded in grey. After a tree reduction (propagation) step, we can evaluate the similarity between two monophonic melodies. Moreover, it has extended the tree representation so that the approach is compatible with polyphonic music, by using multiset labels, where each root defines the set of notes (chords) that have been played during this time interval. Therefore, this approach can deal with both monophonic and polyphonic music.

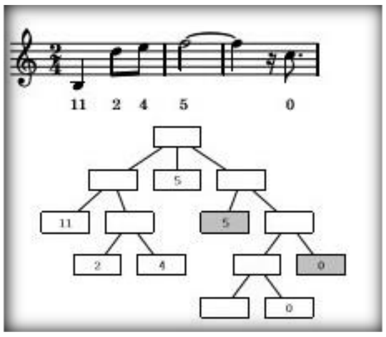


*Figure 2.1-6. Example melody and its corresponding tree data structure representation (Valero & Quereda, 2010).*

The experiment was carried out and compared with existing MIR systems, some of which, such as CBrahams, have appeared in Table 2.1-2. Moreover, CBrahams is the one which can do all three types of the matching except taking audio as the system input. Thus, all the music notes should spread out with the same time frame as the tree structure. Otherwise, during the reduction step for approximate matching, melodies with variations might be reduced to the same abstract melody, as only the important nodes from the leaves nodes are kept onto the higher layer parent nodes. As a consequence of this, the limitation of this approach is that it depends on the meter structure of the input resource and thus, can be very difficult to represent ties, dots and syncopations. Even though, when the tree representation applies to the more complex melodies of polyphonic music pieces, the multiset label cannot distinguish the difference between whether the notes in this time frame come from the right hand part or the left hand part, which will affect the actual similarity scores. Therefore, it is necessary to find a new coding scheme so that more types of music/music notes can be represented/encoded. For example, we can combine Hrušková & Hvolka's vector format (2011) and Valero & Quereda's tree structure (2010).

Finally, Huang et al. (2013) developed the MidiFind system that deals with MIDI files and uses a hybrid system that carries out the music similarity searching from piano music pieces. They have shown the logic structure of MidiFind to illustrate the step of their scalable system. Their system has achieved 99.5% in precision and 89.8% in recall by using machine learning and music analysing. Even with that result, they have suggested using more machine learning techniques. For example, build a 'word' based knowledge system to implement a decision tree to add further control to the existing system. This paper indicates that we need to use various machine learning techniques to compute music similarity.

From those case studies and critiques, here are some critical points about the main features and the major limitation for audio files and symbolic-based MIR systems:

- ✓ For the files used in symbolic-based MIR systems, such as MIDI files, they can be easily stored and require less disk space, as only the core information about the music is required, such as the pitch for the key, the time to press the key and the time to release the key. Hence, they are faster in processing time.
- ✓ As the melody i.e., the flow of the music, has been well represented in various symbolic-based approaches, we can recognize similar melody patterns and carry out the exact and the approximate music content-based search for different musical features. Thus, it is easy to do the task of identifying the original music pieces given its variations.
- ✓ High retrieval accuracy has been obtained for the melody-based search.
- – As the symbolic limits the pitch set and time resolution, music with continuous pitch sets are hard to symbolize which means that not all music can carry out the symbolic-based music search easily. Similarly, lyrics are difficult to incorporate.
- – Music search purely based on the rhythm obtained low retrieval accuracy.

## 2.2. Audio VS Symbolic

From previous sections, we have already literally concluded that the symbolic files, such as MIDI, require less disk space compared to the audio files, and possess limits to the discrete pitch set and time resolution. This means that if we convert the audio to symbolic, we have filtered out certain unimportant musical features and altered certain musical properties, which will lead to a different real-time spectrum for the sound signal of the symbolic music compared to the original audio music (Chen, et al., 2016), if we playback the symbolic files.

Therefore, in this section, we further analyze the differences between the audio and the symbolic files, from the concept of music representation and storage using the case studies.

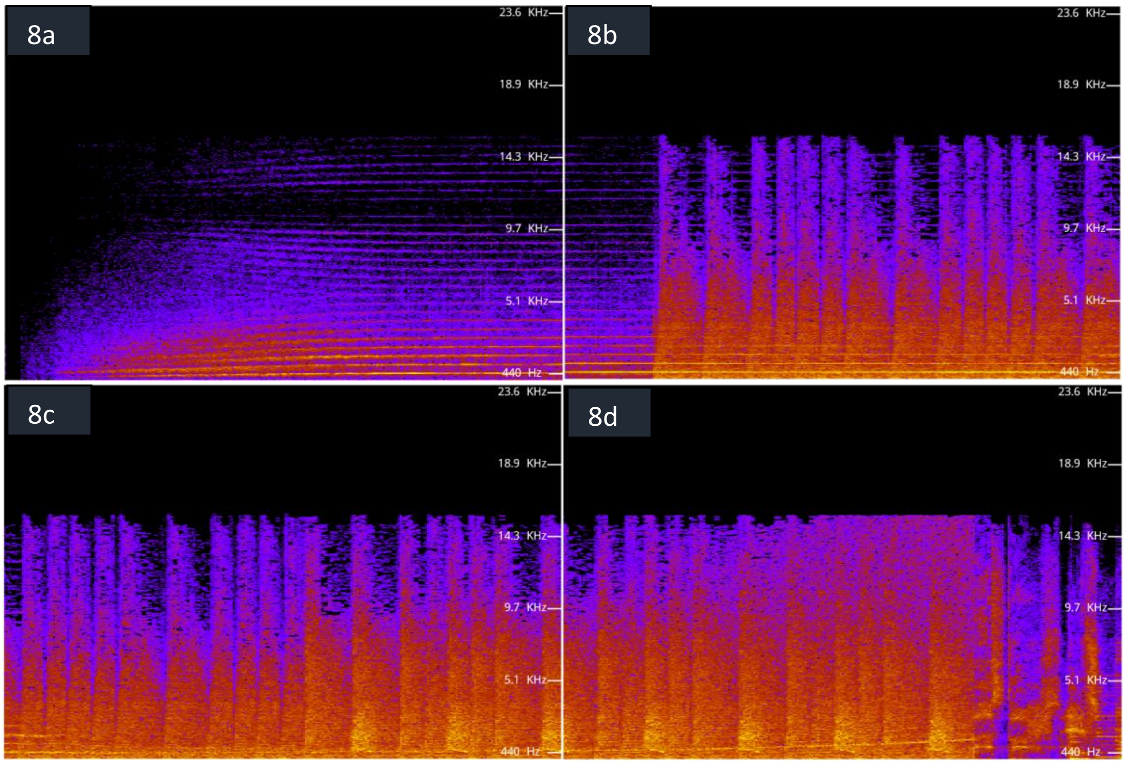


*Figure 2.2-1. (8a-8d) Spectrum Analyzer for the intro of [45]* [1].

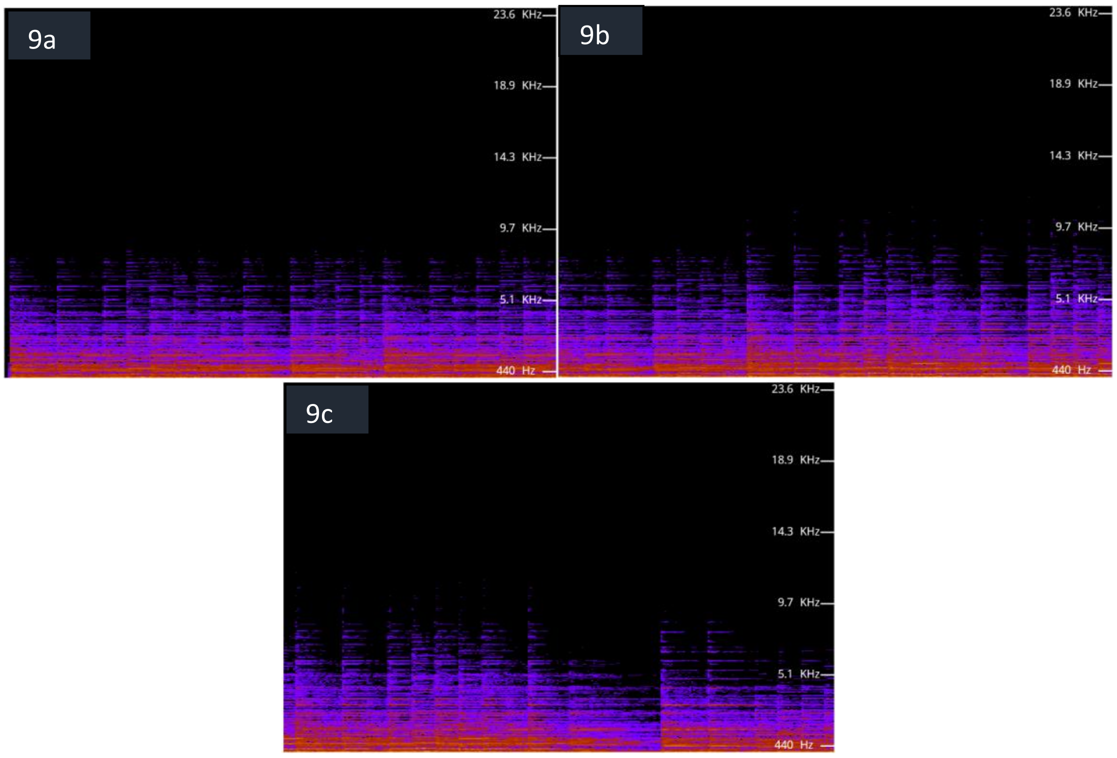


*Figure 2.2-2. (9a-9c) Spectrum Analyzer for the intro of [45a]*[1].

[1] https://academo.org/demos/spectrum-analyzer/

For our case study, we have chosen two versions for the intro part of “Zero For Conduct” by Bastarz (of Block B) [45], as it contains music which consists of both continuous and discrete pitches. The first version is the original song from their official music videos [45], the second is the MIDI version of the piano cover [45a]. We used spectrum analyzer[1] to see the difference between those two versions, as shown in Figure 2.2-1 and Figure 2.2-2 respectively. In both figures, the hidden horizontal x-axis is the time in seconds (*sec*) and the y-axis is the pitch in Hertz (*Hz*). The colour indicates the sound intensity of the frequencies – the brighter the colour, the higher the intensity.

From Figure 2.2-1, we can see that the original music start with a steady-continuous-increase-in-frequency siren-like sound, from approximately E4 to around C5 sharp, followed by the instrumental melodies, before the main vocal melody starts. As the piano only consists of 88 piano keys, the intro for the MIDI version in Figure 2.2-2 does not include the first few seconds of the original intro. Moreover, none of the symbolic version (MIDI, music sheet etc.) that is available on-line can manages to have the siren-like part.

On the other hand, the sound intensity for the MIDI version is on average lower than the sound intensity for the original audio version, the colours in Figure 2.2-1 are much brighter than in Figure 2.2-2 and includes some higher pitches. This was affected by the different timbre between the original recordings and the MIDI piano synthesizer, for example, each instrument has different harmonies and each singer has different tones. However, the fundamental pitch, the tempo and the beat have been matched well among the audio and the symbolic files.

From the storage perspective, the file size for the MIDI file is 6.53 KB, whereas the file size for the MP3 file downloaded from the original music video is 4.51MB, consisting of the following configurations: 44,100 *Hz* stereo, 32-*bit*, and 192 *Kbps* CBR. Despite the fact that the MIDI version does not cover the whole song, but as 6.53 KB is much smaller than 4.51 MB, we can still claim that the symbolic file is, in general, much smaller than the audio file. This can be proven by converting the MIDI file back into an audio file of the same music time length (the MP3 file size is 1.11 MB), as well as from the literature review, e.g., (Holm, et al., 2005). Moreover, analogue audio needs more space than digital audio.

Therefore, we can conclude into the follownig summary table, as shown in Table 2.2-1.

| | Audio | Symbolic |
|---|---|---|
| **Advantages (Features)** | • Able to store and represent sound consisting of continuous pitch set, either analogue or digital format.<br>• Lyrics for the vocal and various tones are representable. | • Small file size on average.<br>• Both human and computer readable and understandable.<br>• Easy to convert back into the audio file. |
| **Disadvantages (Limitations)** | • Large file size in general, and huge for top class sound quality.<br>• Difficult to capture certain melodic information, such as the time and key signature. | • Music limited to the discrete pitch set and lyrics. |

*Table 2.2-1. Summary Table for the Audio VS Symbolic files.*

## 2.3. Automatic Music Transcription and Signal Process

From the summary table in Section 2.2, we can see that it is easy to convert the symbolic file back to the audio file, but difficult to convert the audio file into a 'perfect' symbolic file or equivalent music sheet. This process is normally known as the Automatic Music Transcription (AMT). Thus, we need to review the difficulties in the AMT process.

Therefore, we briefly introduce the modern technologies applied in music, speech, and other related subject areas. These technologies have been grouped into AMT, Signal Processing.

### 2.3.1. Automatic Music Transcription

As automatic music transcription (AMT) is a sub-category of an MIR, where we convert the acoustic audio into some symbolic-based music notation forms (Cogliati, et al., 2016; Fournier-S'niehotta, et al., 2016). Here are the critiques on some AMT related papers.

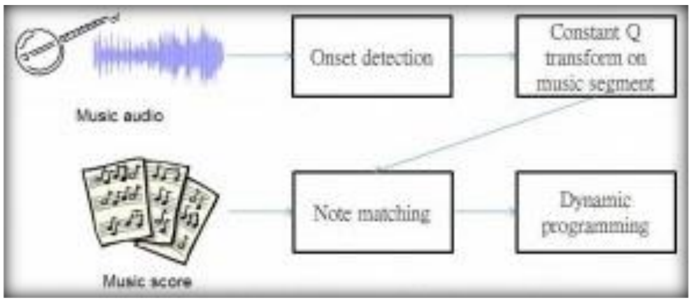


*Figure 2.3-1. Flowchart for the audio-to-score alignment algorithm (Chen, et al., 2016).*

In Demopoulos & Katchabaw's paper (2007), the majority of the paper was discussed music representation and symbolic pattern matching, such as string matching. Most importantly, they claim that the transcription from the audio can be complex depending on the amount of musical information needed. Thus, it introduces two methods in order to get the fundamental frequency and converting it into symbolic music representation, which is known as AMT. This allows audio-based MIR systems to use the symbolic approach, like those green MIR systems in Table 2.1-2, not just using the audio fingerprinting approach. However, this paper claimed that for these kinds of systems, getting the time and key signature information is much harder than getting the pitch and rhythm information. Similar challenges for AMT were claimed in Benetos et. al.'s paper (2013).

On the other hand, Chen et. al. (2016) proposes an innovative method to do the audio-to-score alignment. Their task is to evaluate the matching similarity between the audio recording and the music score. From their flowchart, as shown in Figure 2.3-1, their approach were divided into four parts: onset detection and segmentation, constant Q transform, note matching and dynamic programming. The onset detection stage is covering the beat tracking and polyphonic music transcription, which relates to the challenged tasks mentioned in previous papers. However, the difference is they were using the music score as a guide to verify the audio recording by alignment, rather than generating non-previous-determinated music score. Similarly, Kwon et. al. (2017) used Recurrent Neural Network-based AMT to audio-to-score alignment on the piano performances.

### 2.3.2. Signal Processing

Here are the transformations using several fourier transformation techniques in signal processing that we can use for the music feature extraction stage for an MIR system.

The first typical fourtier transformation is the Short Time Fourier Transformation (STFT). Ning's team (Yang, et al., 2017) used STFT to separate vocal and background music, whereas Simon (Dixon, 2000) applied STFT to recognise note on solo piano music performance of Mozart piano sonatas. According to Simon's paper, he obtained a recognition accuracy of around 70% – 80%, and has emphasised the limitation of using the synthesised data, the evaluation function and the issue with the accuracy of the dynamics and the offset times.

Another typical fourier transformation will be the Fast Fourier Transformation (FFT), which can be applied in Phase Vocoder (Portnoff, 1976), which is a system that provides parametric representation for a speech waveform. As it describes, the representation is in a sequence that has been discretised with optimized approximation. With the similar approach, Nathan's team (Lenssen & Needell, 2014) used FFT when dealing with chord recognition.

As our final case, Sharma and Sardana (Sharma & Sardana, 2016) used another technique, named Mel Frequency Cepstral Coefficients (MFCC), to recognize speech from 'one' to 'nine'. It stores each individual word's features into a database, and separates words from a real-time speech by classify with the pre-recorded output. However, the paper has been pointed out the system is very sensitive to noise and sometimes cannot cope with words which sound similar. For example, 'seven' sound has contained 'one' sound. This speech recognition task will be handy for the lyric-based MIR systems with non-text input.

We have the detailed overview of the theories behind those approaches in Section 2.5.3.

## 2.4. Other Achivements using ML

In this section, other achievements from other MIR sub-categories and music related subject areas using various machine learning techniques, were briefly introduced.

Machine Learning (ML), by definition, is the scientific study of algorithms and mathematical models which automatically learn programs from data. Instead of using manually constructed instructions, ML relies on patterns and inference (reasoning) generalised from data examples (Domingos, 2012). One subfield of ML concerning algorithms inspired by the structure and function through a **complex** deep constructed Neural Network (NN) system with feature learning, is the Deep Learning (DL) (Chen & Lin, 2014). In other words, it is an extension of NNs that accurately assign credit across many **long** casual chains of computation stages. Each computation stage transforms the aggregate activation of the network using either supervised, semi-supervised or unsupervised ML-based algorithms in order to carry out various tasks such as regression, classification and clustering (Schmidhuber, 2015).

Thus, there are many applications from MIR which use ML techniques. For example, Osmalskyj et al. (2012) have applied Artificial Neural Network (ANN) in the music chord recognition for four instruments and achieved reasonable accuracy. Furthermore, they have noticed that the error for piano is still large, and gave a possible reason, the noisy nature of piano sounds. On the other hand, Matityaho and Furst (1995) have obtained 100% success in classifying the music types, whereas Kim et. al. (2013) proposed an expanded Emotion model (XEM) and defined the ontology (MEMO) based on XEM to carry out the music content search using the Korean emotional words, which benefit korean music recommendation systems.

For Natural Language Processing, a subject area which is related to MIR, the combination of Recurrent Neural Networks (RNN) from DL and parallel computing have been used widely in order to deal with large databases (Walder, 2016).

Beyond purely music or sound, Liao et al.'s (2009) used the vector sequence approach to find the association between music/song and its Music Video (MV). The vector sequence denotes the features of the music clip from the segment of sound frequency graph using the Dual-Wing Harmonium model. Since the songs and its corresponding MTV have a high correlation, then by considering the timing of the useful movements from the MTV, the time frame of the music itself can be obtained with higher accuracy using feature learning. Moreover, if there are more than one people singing the song, then by combining the music and the MV, the melody line can be further grouped and acquire extra information from individual singers. This extra information can be used for other applications, e.g., re-performance, in theory, can be identified.

## 2.5. Background Knowledge Overview

So far, we have summarized the benefits and limitations of each audio or symbolic files and analyzed audio and symbolic MIR systems. Majority of audio MIR systems have used collections of wave samples as audio fingerprints to identify which music you are currently listening to, whereas symbolic MIR systems have used pattern recognition from stream sequences and similarity scores to let us search the song from various music features. We have ended up with introducing several techniques which have been applied in music and other related subject areas.

Therefore, before carrying on the methodology and the MUCASM design, this section provides a brief discription of relevant topics, including important definitions, mechanisms and theories, and groups in the following four topics: Machine Learning; Music Notations and Terminologies; Signal Processing and Topology from mathematics.

### 2.5.1. Machine Learning

Due to the size of our dataset which leads to the small scale of feature learning, it is hard to achieve the deep learning level at the start. However, we can still built a mechanism that is DL-upgradable. Thus, in this section, the main concept of some particular machine learning techniques and neural network, which have been used in our MUCASM design (Section 5.1), were briefly described.

#### *2.5.1.1. Decision Tree*

Decision Tree (DT) classifier is one of the classic machine learning approaches that deals with multistage decision making. This mechanism has been successfully used in various areas that need classification, such as handwriting recognition, medical diagnosis etc. (Safavian & Landgrebe, 1991).

For this classical classification technique, it consists of a set of data that is pre-classified into $n$ classes, where $n \geq 2$. A decision tree has *decision nodes* and *leaf nodes*, where the decision node gives some constraint on one attribute and the leaf node outlines the class. Hence, when constructing the DT, we need to know the series of cuts from the root node to a leaf node, such that it can partition the data point. Most of the time, we tend to use a binary cut, which divides the region into two halves (Liu, et al., 2000). Figure 2.5-1 shows an example of a decision tree.

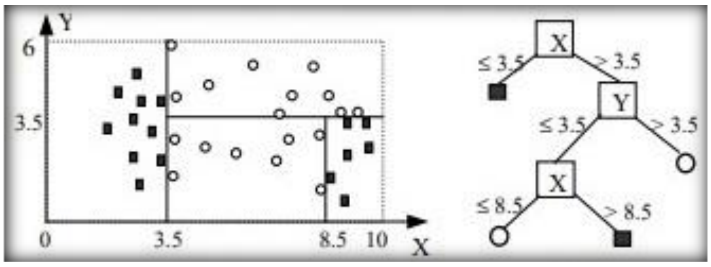


*Figure 2.5-1. An example of Decision Tree (Liu, et al., 2000).*

Moreover, the conditions or rules for each branch of the decision tree can be either numeric (as shown in Figure 2.5-1) or non-numeric. Numeric normally consists of continuous-valued attributes, whereas non-numeric can normally be pre-categorised with fuzziness, or, in other words, categorised into discrete-valued attributes (Tan, et al., 2018).

Instead of partition the data point, we partite the similarity measurement methods in our MUCASM system.

Detailed information about Decision Trees and other basic machine learning techniques were in Tan et al. (2018)'s book.

#### 2.5.1.2. Self-Organizing Map

Self-Organizing Map (SOM) is an unsupervised learning based Artificial Neural Network (ANN). The main idea, as described in Kohonen's 《The Self-Organizing Map》 (Kohonen, 1990), is that the dataset itself acts as a 'teacher' with appropriate hidden neuron update formula. The mechanism is that each hidden neuron rearranges by itself. The hidden neuron with the highest Kohonen map similarity, i.e., whose weight vector lies closest to the input vector, known as the winning neuron (best match unit), determines the final clustering category. Figure 2.5-2 shows two example of architectures of SOM networks. This learning method has been adopted in our system, such that we are able to treat each original music/melody as the hidden neuron. Moreover, it is possible to use clustering algorithms to solve a classification problem (Evans, et al., 2011).

Detailed information about SOM and other neural networks were in the following references: (Kohonen, 1990; Heaton, 2015; Miljkovic, 2017).

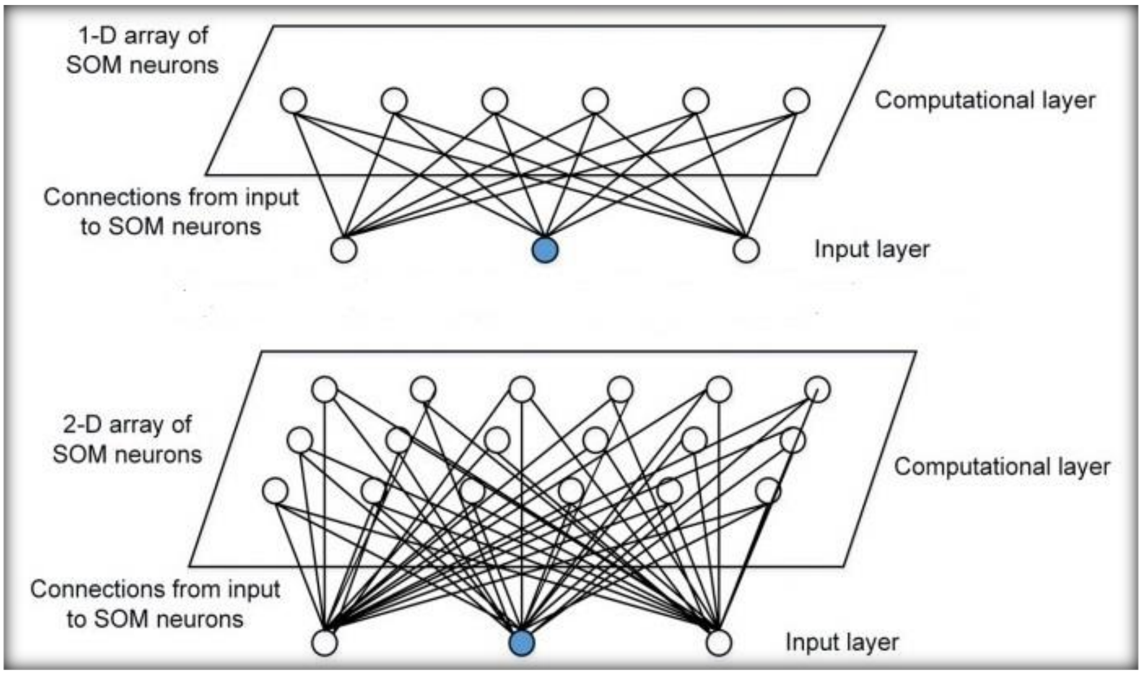


*Figure 2.5-2. Architecture for 1-D and 2-D SOM Neural Networks (Miljkovic, 2017).*

#### 2.5.1.3. Ensembled Learning and Reinforcement Learning

Ensembled Learning (EML), by definition, is a supervised learning that uses various ML algorithms to obtain better performance. When it comes to complex decision making, we weigh different opinions and combine them on our final decision making (Zhang & Ma, 2012). This process is just like if we listen to the advice or informatiom from different experts, which can happen via a human, a textbook, internet or any other media, and combine them into our own practical problem. Our own problem may not be perfectly solved by one single piece of advice, e.g., when the criteria is slightly different. Therefore, EML can deal with large volumes of data, as well as too little data, with complex data boundaries or even through dealing with data fusion (Polikar, 2006).

Figure 2.5-3 shows the relationship between having and not having ensembled learning.

There are various Ensemble Learning algorithms. E.g., Majority Voting, Stacked Generalization etc. (Polikar, 2006; Wiering & van Hasselt, 2008). We applied the Stacked Generalization (Polikar, 2006) idea to our MUCASM system.

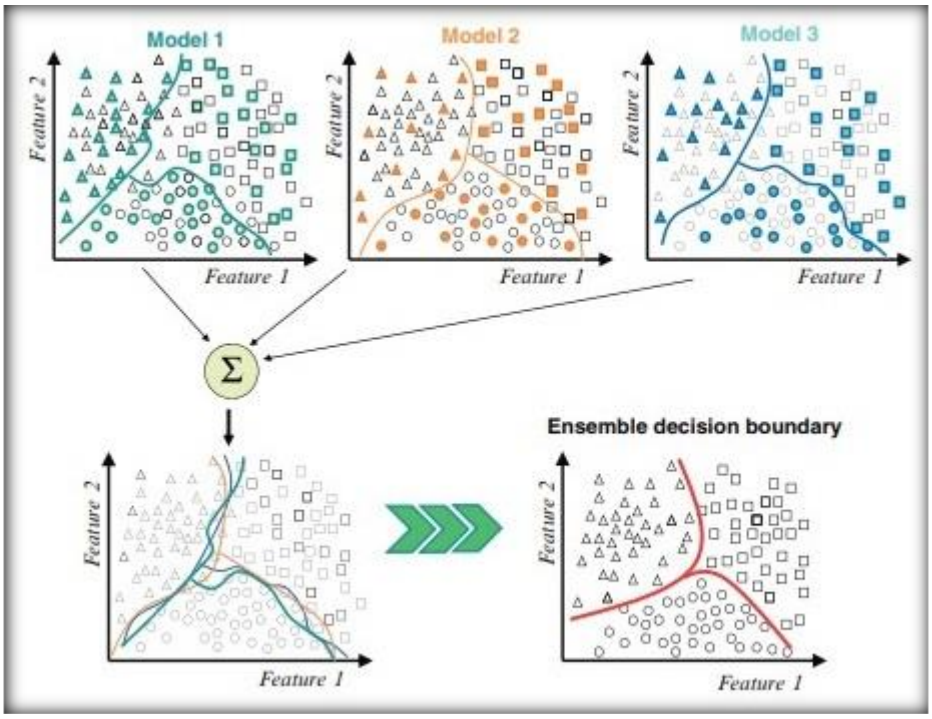


*Figure 2.5-3. Example of Emsembled Machine Learning process (Zhang & Ma, 2012).*

For further details about Ensembled Learning, please refer to Cha's 《Ensemble Machine Learning – methods and applications》 (Zhang & Ma, 2012).

Reinforcement Learning (RL), from its definition, is a learning mechanism that controls agents to take corresponding action(s), from the dynamic environmental state and feedback reward (Sutton, et al., 2018), where theoretically, the reward value for the value-based Reinforcement Learning comes from the environment, known as the interpreter. The learning process can be based on model, policy or value-function. E.g., Q-learning, policy iteration. This mechanism has been widely applied in multi-player games that allows human against A.I., e.g. AlphaGo. Moreover, if we have a complex environment, like in AlphaGo, EML can be combined onto RL (Wiering & van Hasselt, 2008).

Here are the descriptions for those terms related in RL (Sutton, et al., 2018):

- Action (A): the set of all the moves an agent can take.
- Iterative time (t): the iteration count for the current action.
- State (S): current environment situation.
- Reward (R): immediate feedback from the environment to evaluate the last action.
- Policy (π): the strategy that the agent employs in order to carry out the next action.

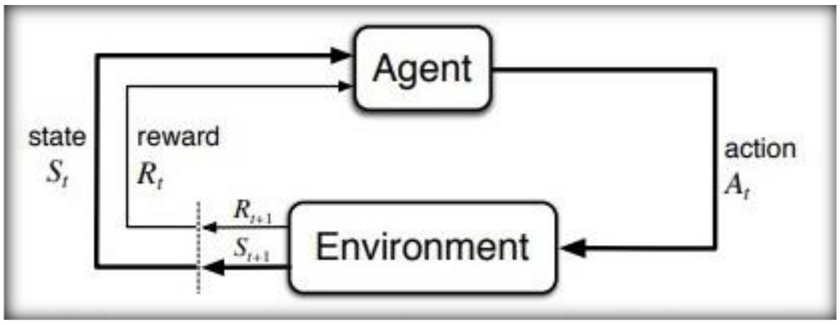


*Figure 2.5-4. Architecture of Reinforcement Learning (Sutton, et al., 2018).*

- Value (V): the expected long-term return, opposed to R.
- $V_\pi(S)$: the expected long-term return of the current state under policy $\pi$.

Hence, we have the architecture of the RL, as shown in Figure 2.5-4.

There are various Reinforcement Learning techniques. 'Comparison between algorithms' is the RL technique which will be adopted in our MUCASM system (Wiering & van Hasselt, 2008).

Detailed information about Reinforcement Learning were in Sutton et al. (2018)'s book.

Moreover, Wiering and Van Hasselt (2008) showed that it is possible to having RL-based EML. Thus, our MUCASM system is one of the RL-based EML. Thus, for the individual classifiers, against various variation types, can be integrated into one Mega-classifier, based on CBA & SG. This allows us to choose which individual classifier is suitable for each variation type.

### 2.5.2. Music Notations and Terminologies

From the point we have mentioned in Section 1.1, the music sheet allows the musicians to easily read, understand, and compose the music, as various musical notations (music symbols) and terminologies have been used, each of which has its own meaning. For example, Figure 2.5-5 shows the circle of fifths. Thus, a summary table of some typical music symbols and musical terminologies has been provided in Table 0-1 (Gerou & Lusk, 1996; Klapuri & Davy, 2006), in alphabetic order.

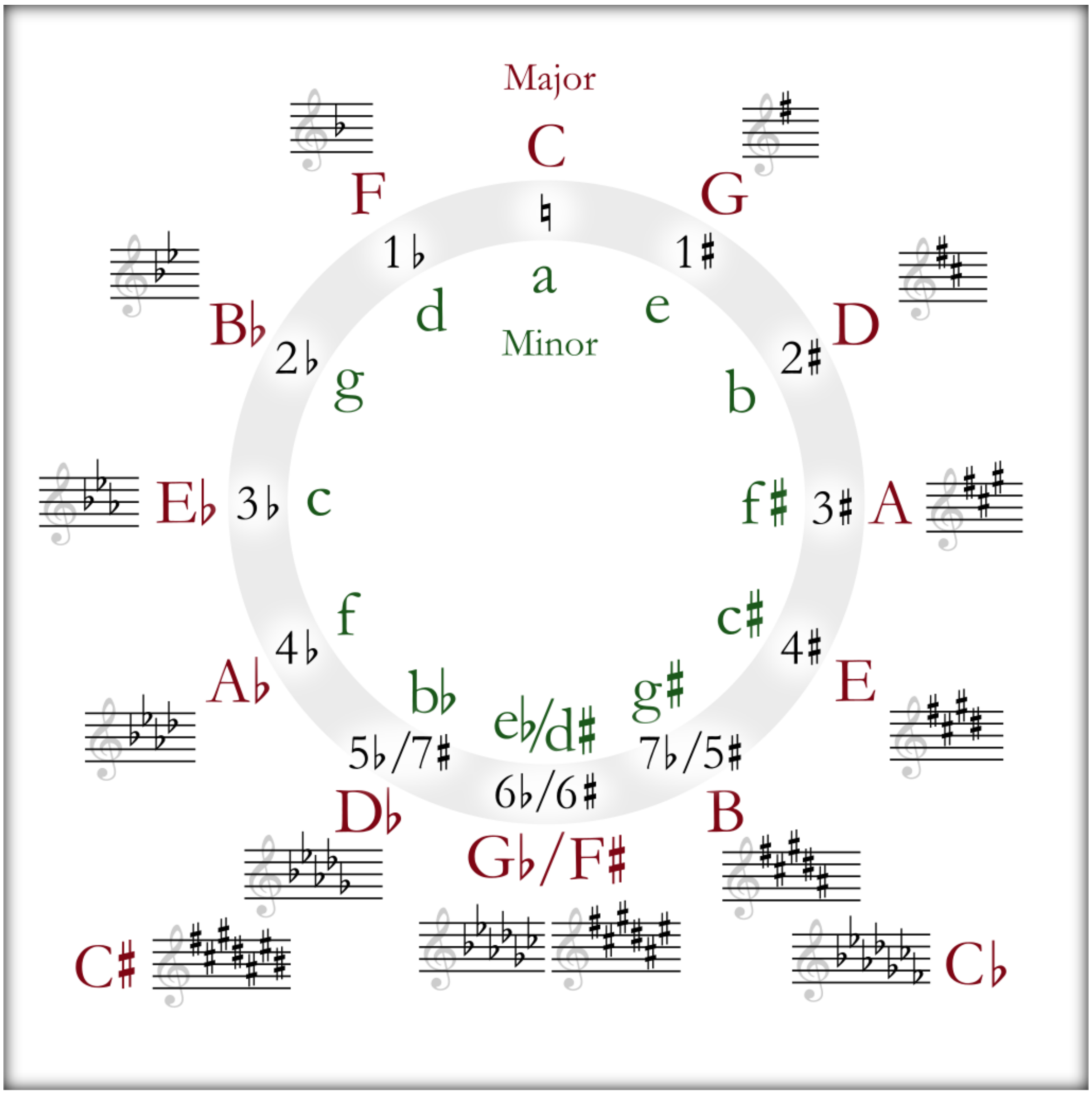


*Figure 2.5-5. Circle of Fifths (Gerou & Lusk, 1996).*

### 2.5.3. Signal Processing

Signal can be known as a function that represents information for a system or an attribute of some phenomenon. There are several formats for a particular signal such as acoustic wave, electromagnetic wave along with many other types (Priemer, 1991). Therefore, more specifically, we are interested in analogue (audio) signal and digital signal in this project. The difference is that one of them is a continuous wave signal with respect to time along the time-domain, and the other is using a sequence of discrete values to represent the original continuous quantities, sampled with time intervals, as shown in Figure 2.5-6.

Signal Processing (SP) is to use certain algorithms or operations that input a signal and output, either the resulting signal or the signal analytical result. For the typical examples of the output being a signal, would be the audio-digital and digital-analog converter, and the filter from the Digital Signal Processing (DSP) (Priemer, 1991; Spanias, et al., 2006; Tan & Jiang, 2018). Moreover, these signals are represented in the time domain.

On the other hand, for the typical example of the output being the analytical result would be the signal spectral analysis, by applying appropriate Fourier Transformation (FT) (Tolstov, 2009; Zainuddin Lubis, et al., 2016). FT is used when representing signals in the frequency domain. First, consider the definition of the fourier series (Kessler, 2007):

*Definition 1. Fourier Series*

*A Fourier Series for a function f is defined as:* :

$$f(x) := \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n \cos\frac{nx\pi}{T} + b_n \cos\frac{nx\pi}{T}\right), n \in \mathbb{Z}^+$$

*where*

$$a_n := \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos\left(\frac{nx\pi}{T}\right)dx \text{ and } b_n := \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin\left(\frac{nx\pi}{T}\right)dx \tag{1}$$

*or alternatively, from the Euler formula, we can re-write as:*

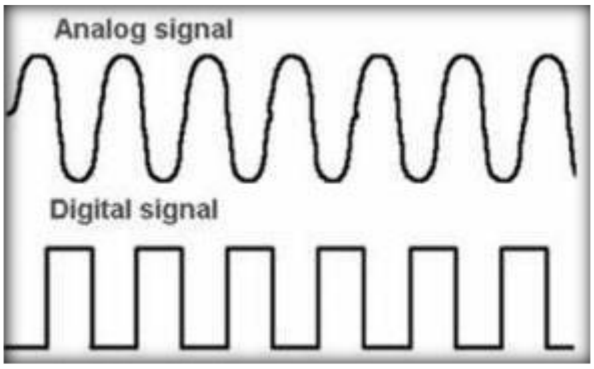


*Figure 2.5-6. Analogue signal and digital signal (Zainuddin Lubis, et al., 2016).*

$$f(x) := \sum_{n=-\infty}^{\infty} C_n e^{inx} \ \text{where}\ C_n = \frac{1}{2T}\int_{-T}^{T} f(t)e^{-n\frac{\pi}{T}t} \tag{2}$$

Hence, the general Fourier Transform would be:

$$F(w) :=\ F\big(f(t)\big) := \int_{-\infty}^{\infty} e^{-i\omega t} f(t)dt \tag{3}$$

whereas the corresponding Inverse Fourier Transform would be:

$$f(t) :=\ F^{-1}\big(F(w)\big) := \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega t} F(w)dw \tag{4}$$

Therefore, for STFT, the main concept is to introduce a convolution-based sliding window function w onto Equation (3), as shown in Equation (5). This can help improving the evaluation of Signal-to-Interference Ratio and Signal-to-Distortion Ratio (Yang, et al., 2017).

$$STFT\{x(t)\}(m,\omega) := \int_{-\infty}^{\infty} x(t)w(t-m)e^{-i\omega t}.dt \tag{5}$$

Moreover, Michael (Portnoff, 1976), Nathan and Deanna (Lenssen & Needell, 2014) have illustrated the connection between STFT and Fast Fourier Transformation (FFT), a form of Discrete Fourier Transformation (DFT), and claim that FFT can save log $N$ versus $N$ per output value. I.e., $O(N^*\log N)$ instead of $O(N^2)$ where $N$ is the dataset size. However, all DFTs were used only for discretised signals. Thus, unlike integration for STFT, we use summation for FFT, which is shown in Equation (6).

$$FFT\{X_k\} := \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{6}$$

where $k = 0, 1, \ldots, N$-1.

Furthermore, if we are using Equation (7), then we can map the powers of the spectrum derived from the Fourier Transformation onto the ‘Mel’ scale (Sharma & Sardana, 2016). This is known as the Mel Frequency Cepstral Coefficients (MFCC).

$$Mel(f) := 2595 * \log_{10}\left(1 + \frac{f}{700}\right) \tag{7}$$

In conclusion, STFT, FFT and MFCC are typical signal processing techniques which can be applied when analysing musical, speech and other sound related data, depending on whether we are dealing with continuous or discrete sound waves. Thus, FFT is choosen in Section 4.5 as we have sampled the sound sinusoidal value. Detailed information about signal processing were in the following references: (Priemer, 1991; Tolstov, 2009).

### 2.5.4. Topology

Topology is a mathematical field which studies different sorts of geometrical space under continuous transformation or deformation (Hatcher, 2002). Therefore, it mainly uses mathematical terms like ‘homotopy’, ‘homeomorphism’ and ‘isomorphism’, to find relationships and similarities between the two objects, under the topological space. The most famous example is that a normal coffee mug is homeomorphic (similar) to a donut ring, both known as a torus, and is dissimilar to a ball, known as a sphere.

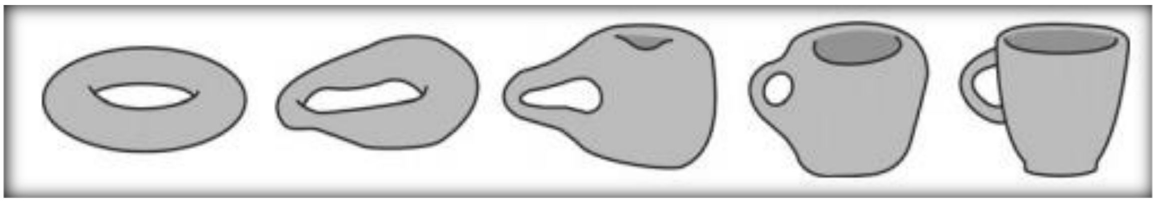


*Figure 2.5-7. Topological transformation from a donut ring to a coffee mug (Coelho & Zigelbaum, 2011)*

Figure 2.5-7 shows the topological transformation from a donut ring shape to a coffee mug, in order to illustrate the continuous transformation between the two homeomophic objects (Coelho & Zigelbaum, 2011).

For the concept of two objects being 'similar' to each other under topological spaces, we have the following definition:

*Definition 2. Homeomophism*

*A function $f: X \to Y$, between two topological spaces is homeomophism iff f is a bijection and both $f$ and $f^{-1}$ are continuous (cts).*

Hence, given two distinct points $(v_s, v_t)$, with two different continuous paths joining these two points, only the same starting point $(v_s)$ and the same finishing point $(v_t)$ matters. We can find a one to one invertible mapping, such that for any arbitrary time between $(v_s)$ and $(v_t)$, any single point from one path can be mapped to a point from the other path (Farber, 2003). In other words, for $v_s, v_t \in p_1(t), p_2(t)$ where both $p_1, p_2$ are functions maps $X := I \in [0,1]$ to the space $Y$, we have $p_1(0) = p_2(0) = v_s, p_1(1) = p_2(1) = v_t$ and there exists an invertible cts function $f$ for all of $t \in [0,1], f\big(p_1(t)\big) = p_2(t)$ & $f^{-1}\big(p_2(t)\big) = p_1(t)$. Sometimes, this is also known as isomorphsim.

If we treat these paths as two functions, instead of pointwise, then this leads to another definition named homotopy:

*Definition 3. Homotopy*

*Given two continuous functions f and g from topological space X and Y, then f and g are homotopic, denote as $f \simeq g$, iff $\exists H: X * [0,1] \to Y$ s.t. $H(x,0) = f(x)$ and $H(x,1) = g(x)$ $\forall x \in X$. Thus, H is a homotopy between f and g.*

This leads to a definition of homotopic equivalence:

*Definition 4. Homotopic Equivalences*

*For a continuous map $f: X \to Y$, if there exists a g s.t. $g^\circ f \simeq I_X$ & $f \circ g \simeq I_Y$, where $I$ is the identity, then we say X and Y were homotopic equivalent and f and g were homotopic equivalences, where g is the homotopy inverse of f.*

Hence, we can conlude that every homeomorphism is a homotopy equivalence, but not the converse, as homotopic equivalences did not guarantee that $g$ is exactly $f^{-1}$.

For a quick proof of the most famous example, using the above descriptions, any torus shape has one hole so that each point can form two 'perpendicular' circular paths that intersect only once at the point itself. I.e., for two cts circular paths $p_1$ $and$ $p_2$ generated from the point $v$, we only have one intersect point at $p_1(0) = p_2(0) = v_s = p_1(1) = p_2(1) = v_t = v$. Thus, we can derive that a coffee mug is homeomorphic to a donut ring. However, for the sphere shape, two intersect points can be formed. I.e., not only having $p_1(0) = p_2(0) = v_s = p_1(1) = p_2(1) = v_t$, but also having $p_1\left(\frac{1}{2}\right) = p_2\left(\frac{1}{2}\right)$. If we remove the point $v$, then $p_1$ $and$ $p_2$ become disconnected for the torus, whereas $p_1$ $and$ $p_2$ remain connected for the sphere. In other words, the mapping function H from $p_1 + p_2$ $from$ $f_{torus}$ to $p_1 + p_2$ $from$ $g_{ball}$ is injective, but not surjective:

$$H\left(\left(p_1\left(\frac{1}{2}\right)\right), 0\right) = f_{torus}\left(p_1\left(\frac{1}{2}\right)\right) \to H\left(\left(p_1\left(\frac{1}{2}\right)\right), 1\right) = g_{ball}\left(p_1\left(\frac{1}{2}\right)\right)$$

$$H\left(\left(p_2\left(\frac{1}{2}\right)\right), 0\right) = f_{torus}\left(p_2\left(\frac{1}{2}\right)\right) \to H\left(\left(p_2\left(\frac{1}{2}\right)\right), 1\right) = g_{ball}\left(p_2\left(\frac{1}{2}\right)\right)$$

$$f_{torus}\left(p_1\left(\frac{1}{2}\right)\right) \neq f_{torus}\left(p_2\left(\frac{1}{2}\right)\right) \ but\ the\ target\ g_{ball}\left(p_1\left(\frac{1}{2}\right)\right) = g_{ball}\left(p_2\left(\frac{1}{2}\right)\right)$$

Therefore, the torus is not homeomorphic to the sphere.

Figure 2.5-8 illustrates the difference between the torus and the sphere.

As Figure 2.5-8 shown, the blue circular line is $p_1$ and red circular line is $p_2$, these only intersect at one point, v, for the torus, but two points, v and w, for the sphere.

Detailed information about topology were in the following references: (Hatcher, 2002; LaValle, 2006).

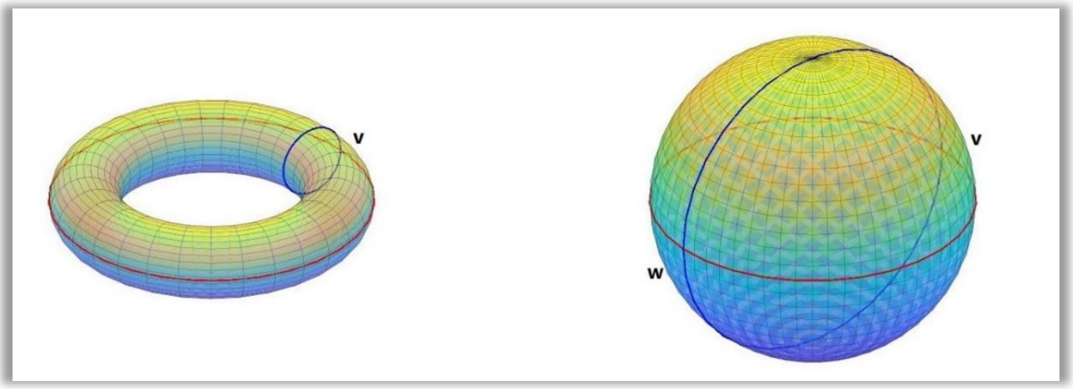


*Figure 2.5-8. Two perpendicular circles each for both Torus and Sphere. Created in MATLAB.*

All these concepts (definitions, theorems, and proofs) from the topology overviewed above, will be useful when dealing with the reasoning of the MML design in Section 4.2.

## 2.6. Summary

From Section 2.1 to 2.4, we have critically evaluated the existing music storage, music representation and music information retrieval systems. We found out it is necessary to combine both audio and symbolic approaches to retain each approach’s advantages. In other word, we need to find a symbolic approach such that it is easy to store, access, manipulate, and most importantly, the music representation for the new approach is closer to the audio approach compare to existing symbolic approaches. Thus, with this new approach, we can carry out the following task: identify the original music piece from its musical variation using both audio fingerprint and symbolic pattern matching, so it can be used in those applications or scenarios described in Section 1.1.

In order to achieve this, we have further overviewed some relevant backgound knowledges in Section 2.5. These relevant knowledges will appear in the rest of the thesis.

Therefore, in Chapter 3, we start to design an architecture based on those findings and knowledges, in an abstract level.

# 3. Methodology and Architecture Design

Based on those issues for the existing symbolic coding scheme and the MIR systems and relevant background knowledge found by carrying out both primary and secondary researches, we have designed a new architecture, named E$^3$MSD, to make an alternative approach for the Original Music Identifier (OMI), such that the positives were retained and the negatives were addressed.

From those positive and negative findings for audio and symbolic approaches reviewed in Chapter 2, we have derived the following concepts for the alternative approach of OMI: it must include an alternative approach for the symbolic coding scheme and the OMI itself with a appropriate machine learning technique. Detailed feature requirements for the symbolic coding scheme (F1.1 - F1.4), as well as the data structure, are listed below:

F1.1. It improves the expressiveness for the symbolic representation of the audio music pieces

F1.2. It allows the audio and symbolic conversion

F1.3. It allows various similarity measurements between songs, based on the similarity scores, higher accuracy and efficiency can be obtained for the original music identifier

F1.4. It is extendable to the music search engine and plagiarism detection applications.

Detailed feature requirements for the OMI (F2.1 - F2.4) are listed below:

F2.1. It is able to identify the original tracks from the variation.

F2.2. It should consider and balance the contour melody similarity and rhythmic similarity.

F2.3. It should combine the search by the symbolic similarity score with the audio fingerprint.

F2.4. It allows different variations to use different methods when identifying its origin.

In order to achieve these, we proposed one new architecture named E$^3$MSD, Expressive, Efficient, and Extendable Music Similarity Detection. Figure 3-1 shows the overall workflow for the E$^3$MSD.

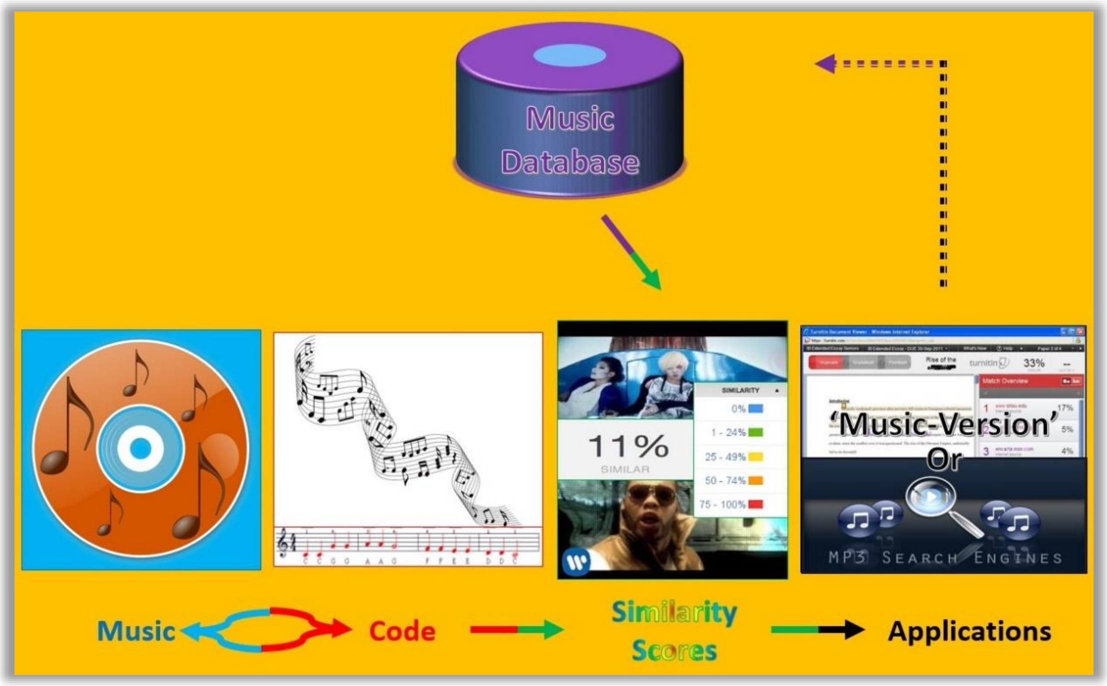


*Figure 3-1. Overall workflow for the E$^3$MSD.*

As shown in Figure 3-1, 'Music' indicates the music is in the audio form and 'Code' in the symbolic form. By comparing the 'Code' from the query against the 'Music Database', we can calculate 'Similarity Scores', which gives us a guideline to several 'Applications'. Depending on the type of application, we may have some feedback for the current music database to update with any relevant application outcomes.

At the 'Code' stage, we will design and define the alternative coding scheme (data structure) such that it combines the features from audio and symbolic files and codes the music in a better way, i.e., to cover F1.1 listed on page 41. From Chapter 2, we can see that, for the existing symbolic coding, such as MIDI files, the following code tuple is always used. The tuple includes a musical note's frequency/notation, its starting time, ending time and its velocity which determines the volume, where the frequency is kept unchanged during consecutive tuples. I.e., the frequency is in the form of a discrete step. Moreover, the new version of MIDI files, including the HD-MIDI, can accommodate more notes' effects from different varieties, such as the vibrato note, by increasing the number of channels and signal controllers. However, this approach does not solve the issue completely (Damm, 1993; Clarke, 2004; Saxena, et al., 2018). Therefore, our coding scheme, should include two different 'languages' which describe the different features of the music. One of the ‘languages’ represents what notes should be played and the other one models how the notes were played. Hence, one of them is corresponding to the existing MIDI files, and the other one tries to shape the audio waves to a stream file, which addresses the limitations of existing symbolic files with a better approximation of the continuous change in frequency and amplitude. In addition, as audio is hard to compute content-based similarity, thus, by separating it into two ‘languages’, we can filter the one that shapes the audio waves and carry out various content-based tasks, which addresses the limitation of the existing audio-based MIR systems. The ‘Code’ will be named Music Definition Language and Music Manipulation Language, which is defined in Section 4.1 - 4.2.

The double headed arrow between ‘Audio’ and ‘Code’ in Figure 3-1 is the conversion between the audio form and the symbolic form, which covers F1.2 listed on page 41. From ‘Code’ to ‘Audio’, we need to examine that the code we have proposed is playable, whereas from ‘Audio’ to ‘Code’, we need to examine that the proposed code is generable. With these two properties, we have further confirmed that the proposed code has satisfied the requirements from the music representation and music storage perspectives and it is ready to move on to music application’s perspective (F1.3 ~ F2.4, page 41). The design and the results for the bidirectional conversion are in Section 4.3 - 4.5.

With the proposed coding scheme, we should be able to compare the music track from the input of the application and any other music track from the music database, and generate a similarity score, as required by F1.3 and F1.4. This similarity score contains a set of similarity scores which is based on different content-based scenarios, e.g., rhythm-based, and thus, several numerical measurements between two music tracks can be provided which in turn, porivdes us with a guideline. In order to achieve this, a self-supervised ensembled learning method has been developed to improve the performance of the Original Music Identifier (OMI) given one of its variations as an input. Self-supervised suggested that the label is automatically generated from the data itself whereas ensembled learning follows from the definition in Section 2.5.1. This learning method has been named as MUsic Classification And Similarity Measurement (MUCASM), which will be act as the core function (OMI) for various applications. It is included inside the arrows around the ‘Similarity Scores’ in Figure 3-1. The detailed design for the self-

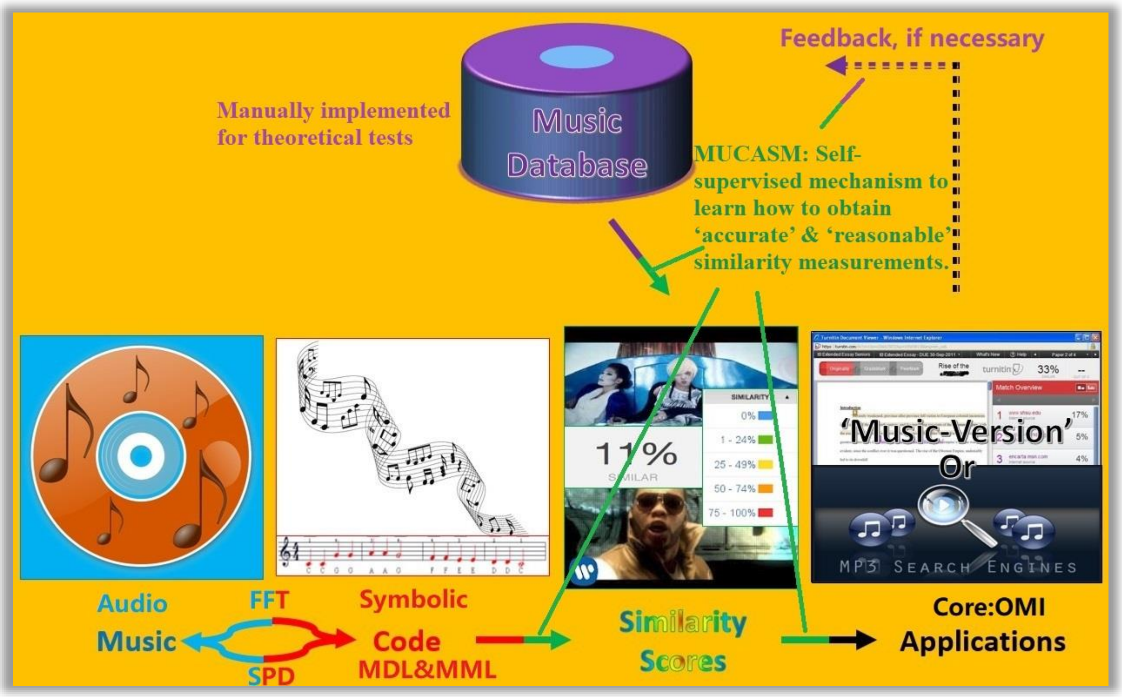

Feedback, if necessary
Manually implemented for theoretical tests
Music Database
MUCASM: Self-supervised mechanism to learn how to obtain 'accurate' & 'reasonable' similarity measurements.
'Music-Version' Or
MP3 SEARCH ENGINES
Audio Music
FFT
SPD
Symbolic Code MDL&MML
Similarity Scores
Core:OMI Applications

*Figure 3-2. Detailed workflow for the E³MSD.*

supervised MUCASM were described in Section 5.1, followed by the results detailed in Section 5.2.

Figure 3-2 is a slightly detailed overall workflow for the E$^3$MSD, such that inside our architecture E$^3$MSD, MDL&MML involves the data type design and MUCASM involves the systems design.

As this is a theoretical project, we need to have some functional and benchmarking tests to the Audio-Symbolic birectional conversion and the self-supervised mechanism for the E$^3$MSD. Thus, we manually create some music tracks in MDL and MML (MDL&MML) format for our 'Music Database' as some variationals might not even exist but we need them for our tests, and based on Figure 3-2, we have divided our project into the following four major stages with a list of functional testings and testing requirements for each individual stage. Each requirement corresponds to a feature/features listed on page 41.

Stage.1. Code to Music

- The possibilities that the new code(s) can represent, includes the continuous glissando (F1.1)

- The performance of the new code(s) playback into audio, using Spectral Pitch Display (F1.1, F1.2)

Stage.2. Music to Code

- The possibilities and the performance of generating the code(s) from the audio file (F1.2)

Stage.3. Code to Similarity Scores

- The possibilities of evaluating the similarity scores from the code(s) using various audio and symbolic approaches (F1.3; F2.2, F2.3)

Stage.4. Applications

- The result on the OMI and compared with some existing examples (F1.4; F2.1, F2.4)

Furthermore, we have made a choice for the overall software development life cycle model, which is the prototyping model from the evolutionary model (Definition 14 in Appendix A) such that we can implement several prototypes (not a whole system) to demonstrate the core ideas from those main stages of $E^3$MSD. The design of those prototypes will be inside the relevent sections: 4.3; 4.5; and 5.1. In addition, we have made the decision to use MatLab when implementing those prototypes as MatLab has several free build-in functions for us to use. All the relevant details are in Appendix A.

# 4. Music Definition Languange and Music Manipulation Language

In this Chapter, we are going to define the exact code(s), followed by two 'systems' which demonstrate the requirements from Stage.1 and Stage.2 in Chapter 3.

## 4.1. Music Definition Language

As described in Chapter 3, we, first, need to define a 'language' that incorporates the basic information/instructions about the musical notes that should be played, such that it has a similar concept to the standard symbolic MIDI files (Rothstein, 1995). Therefore, we define this kind of 'language' as Music Definition Language (MDL), which from the linguistic meaning, 'defines' the music. Moreover, this has a similar purpose to the DDL from Structured Query Language (SQL) (Elmasri & Navathe, 2014).

For computation purposes, MDL is designed to be in a vector format with all numeric values. Furthermore, as MDL corresponds to the symbolic files from the initial design, the following features of a single note should be incorporated into our symbolic-based MDL format (Lubiw & Tanur, 2004; Valero & Quereda, 2010; Walder, 2016):

- The position of the musical note in a modulo 12 system, $N_{mod\ 12}$. For example: Note A is '0', A sharp (or B flat) is '1', and so on. This is illustrated in Figure 4.1-1.
- The set (index) number, $S_I$. This identifies which (modulo 12) set the note lays in. For example: middle C ($261.626\ Hz$) lays in set '0', upper C lays in '1', and so on. However, for computation purposes, the set from A to G# have been treated as the same set rather than the traditional notation which is from C to B. This feature is included in Figure 4.1-1.
- The relative amplitude, $A_d$. This indicates the volume of the note, according to the dynamics.
- The bar number, $B$. In terms of the bar number from the music sheet, this will distinguish the musical notes that appears in different bars.
- The beat time number $B_T$. This refers to the specific beat time, the musical note is played inside the bar number.
- The duration of the musical note $D$. This is a relative time value which relates to the time signature.

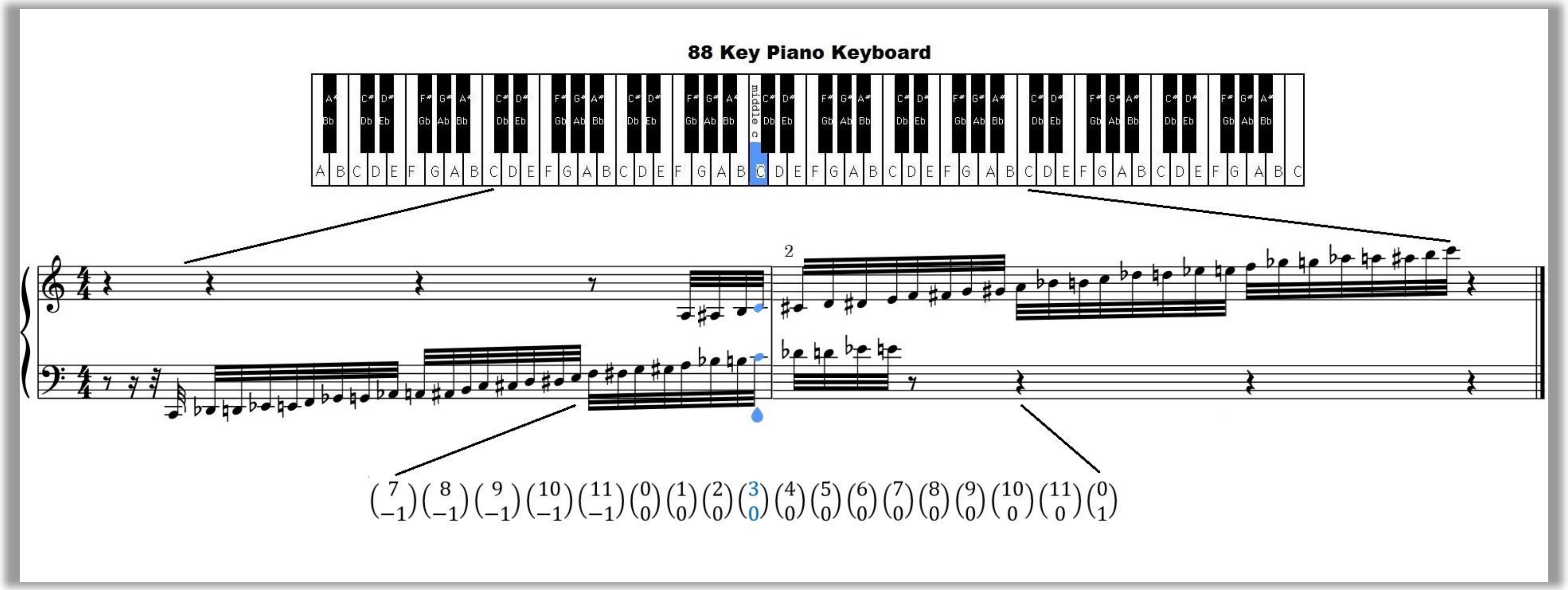

88 Key Piano Keyboard
middle c
$\binom{7}{-1}\binom{8}{-1}\binom{9}{-1}\binom{10}{-1}\binom{11}{-1}\binom{0}{0}\binom{1}{0}\binom{2}{0}\binom{3}{0}\binom{4}{0}\binom{5}{0}\binom{6}{0}\binom{7}{0}\binom{8}{0}\binom{9}{0}\binom{10}{0}\binom{11}{0}\binom{0}{1}$

*Figure 4.1-1. Piano Keys, Piano Sheets and S-MDL.*

Note 1: The vectors were using the format of: $\binom{N_{mod\ 12}}{S_I}$.

Note 2: The middle C has been highlighted in blue.

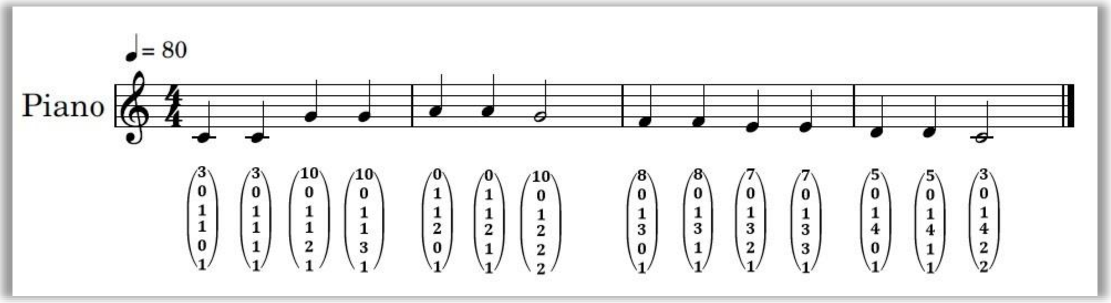


*Figure 4.1-2. Music Sheet and S-MDL for the first four bars of [27].*

To conclude, the first two features together define the note we should play, followed by the relative dynamic value. The next two identify the beginning time of the note. The final feature represents how long the note should be held. Therefore, we have the following mathematical definition for the symbolic based MDL.

*Definition 5. Symbolic-based Music Definition Language (S-MDL).*

*Symbolic-based Music Definition Language (for a music melody) is a stream of vector sequences which describes the (symbolic) flow of the music. Each $6 \times 1$ vector has stored the main six-tuple features of any single musical note in the melody sequence, relates to the music notations. S-MDL can be expressed as follows:*

$$S\text{-}MDL_{Melody} := (S\text{-}MDL_{Note})_i := \left( \begin{pmatrix} N_{mod\ 12} \\ S_I \\ A_d \\ B \\ B_T \\ D \end{pmatrix} \right)_i , \qquad i \in \mathbb{N}^+ \tag{8}$$

*where* $N_{mod\ 12} \in \{0, 1, \dots, 11\}$*,* $S_I \in \mathbb{Z}$*,* $A_d, B_T, D \in \mathbb{Q}^+$ *and* $B \in \mathbb{N}$*.*

For example, Figure 4.1-2 shows the linkage between the music sheet and the S-MDL$_{\text{Melody}}$ of a simple melody, “Twinkle, Twinkle, Little Star” [27].

However, like the existing symbolic codings, similar limitations exist for our S-MDL. The most important one is that the first two entries of the S-MDL belongs to the integer number. I.e., $N_{mod\ 12} \in \{0, 1, \dots, 11\} \subseteq \mathbb{N} \subseteq \mathbb{Z}$ and $S_I \in \mathbb{Z}$. Thus, we define Audio-based Music Definition Language (A-MDL) as containing the following features:

- The fundamental frequency of the sound, $f_p$. Each musical note has its own fundamental frequency (pitch), thus allowing it to extend from the discrete frequency set.
- The amplitude of the sound, $A$. Unlike S-MDL, it is unaffected from the dynamic notations. However, this can vary due to the volume, hence we treat $A = 10 * A_d$.
- The beginning time plays, $t$. By default, it will last until the next A-MDL. Thus, we do not need to code the duration nor the ending time.

These three fundamental features can define a simple sound wave, as all simple soundwaves can be generated from the following sinusoid equation to get its audio waveform (Guillaume, 2006; Kessler, 2007).

$$f(t) = A\sin\left(2\pi f_p t + \varphi\right) \tag{9}$$

where $\varphi$ is phase in radians. In our case, $\varphi = 0$.

Hence, we know what pitch we are making, its volume, and the starting time. The ending time is when we see the next A-MDL, either corresponding to another note, or a rest, where in this case, $f_p = 0$. Therefore, we have the following mathematical definition for the audio based MDL.

*Definition 6. Audio-based Music Definition Language (A-MDL).*

*Audio-based Music Definition Language (for a music melody) is a stream of vector sequences which describes the flow of the music in its acoustic waveform (sinusoidal wave equation). Each $3 \times 1$ vector has stored the main three-tuple features from the sinusoidal wave equation (9). Thus, A-MDL can be expressed as follows:*

$$A\text{-}MDL_{Melody} := (A\text{-}MDL_{Note})_i := \left(\begin{pmatrix} f_p \\ A \\ t \end{pmatrix}\right)_i, \qquad i \in \mathbb{N}^+ \tag{10}$$

*where $f_p, A, t \in \{0\} \cup \mathbb{R}^+$.*

Using the following equations, we can transfer between the 88 piano key number, $n_p$, and its corresponding frequency, $f$, in *Hertz* (Bello, et al., 2000).

$$f = 440 * 2^{\left(\frac{n_p - 49}{12}\right)}$$

(11)

$$n_p = 12 \log_2 \left(\frac{f}{440}\right) + 49$$

(12)

Similarly, for the key numbers in MIDI, $n_M$, simply replace the 49 in Equation (11) and (12) with 69 (Viitaniemi, et al., 2003). This indicates without 49 (or 69 for MIDI), all frequencies have been converted into the key number, $n$, such that the A note with 440 Hz is defined to be the key number zero. Therefore, Equation (13) and (14) illustrate the conversion between our $N_{mod\ 12}$ and $S_I$ from S-MDL and $f_p$ from A-MDL.

$$f_p = 440 * 2^{\left(\frac{12(S_I - 1) + N_{mod\ 12}}{12}\right)}$$

(13)

$$\begin{cases} N_{mod\ 12} = \left(12 \log_2 \left(\frac{f_p}{440}\right)\right) mod\ 12 \\ S_I = \frac{12 \log_2 \left(\frac{f_p}{440}\right) - N_{mod\ 12}}{12} + 1 \end{cases}$$

(14)

On the other hand, according to the tempo measurement in Figure 4.1-2, known as metronome marks (Joutsenvirta & Perkiömäki, 2010), there are 80 crotchets per minute. Thus, one crotchet lasts $60/80 = 0.75$ seconds.

Therefore, Figure 4.1-3 shows the music sheet and its corresponding A-MDL$_{\text{Melody}}$, for the same track [27].

Note: frequencies were rounded to three decimal places.

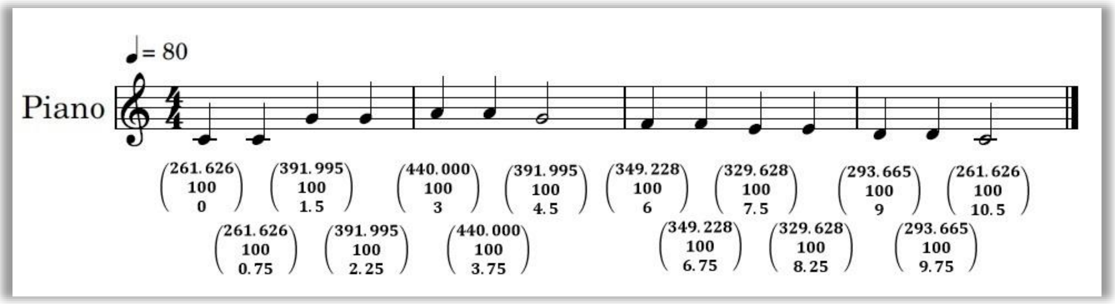

♩= 80
Piano
$\begin{pmatrix}261.626\\100\\0\end{pmatrix}$
$\begin{pmatrix}261.626\\100\\0.75\end{pmatrix}$
$\begin{pmatrix}391.995\\100\\1.5\end{pmatrix}$
$\begin{pmatrix}391.995\\100\\2.25\end{pmatrix}$
$\begin{pmatrix}440.000\\100\\3\end{pmatrix}$
$\begin{pmatrix}440.000\\100\\3.75\end{pmatrix}$
$\begin{pmatrix}391.995\\100\\4.5\end{pmatrix}$
$\begin{pmatrix}349.228\\100\\6\end{pmatrix}$
$\begin{pmatrix}349.228\\100\\6.75\end{pmatrix}$
$\begin{pmatrix}329.628\\100\\7.5\end{pmatrix}$
$\begin{pmatrix}329.628\\100\\8.25\end{pmatrix}$
$\begin{pmatrix}293.665\\100\\9\end{pmatrix}$
$\begin{pmatrix}293.665\\100\\9.75\end{pmatrix}$
$\begin{pmatrix}261.626\\100\\10.5\end{pmatrix}$

*Figure 4.1-3. Music Sheet and A-MDL for the first four bars of [27].*

Further from the example, we can derive the following lemma.

*Lemma 1. The A-MDL and S-MDL Conversion for monophonic melody.*

*Given the standard amplitude, tempo measurement and the time signature, every S-MDL for a monophonic melody can be converted into A-MDL. The opposite direction holds iff $log_2\left(\frac{f_p}{440}\right) \in \mathbb{Z}$ & $t \in \mathbb{Q}$.*

**Proof:**

Firstly, Equation (13) and (14) shows the conversion between the $N_{mod\ 12}$ and $S_I$ from S-MDL and the $f_p$ from A-MDL. Since $S_I \in \mathbb{Z}$ by defnition, then we need $\log_2\left(\frac{f_p}{440}\right) \in \mathbb{Z}$ for Equation (14). Moreover, a rest means a note with *0 Hz* ($f_p = 0$), no sound. Thus, in theory, $\begin{pmatrix}N_{mod\ 12}\\S_I\end{pmatrix} = \begin{pmatrix}0\\-\infty\end{pmatrix}$ is the rest. However, if we provide a filter such that any frequency below a value will be filtered out, i.e., treated as a rest, then we can simply define the corresponding S-MDL as a rest, in other words, $f_p \neq 0$ & $\begin{pmatrix}N_{mod\ 12}\\S_I\end{pmatrix} \neq \begin{pmatrix}0\\-\infty\end{pmatrix}$ for a rest.

Secondly, $A = k * A_d$ for some non-zero constant $k$, can convert the relative amplitude into the real amplitude with respect to the standard amplitude, and vice versa. In our case, $k = 10$. Moreover, for a rest, $A = k * A_d = 0$.

Finally, Equation (15) and (16) shows the conversion between the $B, B_T$ and $D$ from S-MDL and the $t$ from A-MDL, given the duration of a standard note from the tempo measurement (metronome marks) and the time signature. More specifically, $x$ beat per minute means $60/x$ seconds for one relative duration length, $\frac{1}{l}$, and define $D = 1$ for one standard duration length, $\frac{1}{\beta}$. The time signature is given in the form of $\frac{\alpha}{\beta}$. Note both $\beta$ and $l$ were in powers of 2. Thus, for perfectly on time, t ∈ ℚ ⊆ ℝ.

$$\begin{cases} t = \left(\frac{60}{x}\right) * \left(\frac{l}{\beta}\right) * \left((B-1) * \alpha + B_T\right) \\ or \\ t_i = \left(\frac{60}{x}\right) * \left(\frac{l}{\beta}\right) * \sum_{j=0}^{i-1} D_j, \forall i \in \mathbb{N}^+ \; for\ monophonic\ melody \end{cases} \tag{15}$$

$$\begin{cases} B = \frac{t * \left(\frac{x}{60}\right) * \left(\frac{\beta}{l}\right) - B_T}{\alpha} + 1 \\ B_T = \left(t * \left(\frac{x}{60}\right) * \left(\frac{\beta}{l}\right)\right) mod\ \alpha \\ D_i = (t_{i+1} - t_i) * \left(\frac{x}{60}\right) * \left(\frac{\beta}{l}\right), \forall i \in \mathbb{N}^+ \; for\ monophonic\ melody \end{cases} \tag{16}$$

where $x, l, \alpha, \beta \in \mathbb{N}^+$ $and$ $D_0 = 0$.

■

*Lemma 2. The A-MDL and S-MDL Conversion for polyphonic music.*

*Given the standard amplitude, tempo measurement and the time signature, every S-MDL can be converted into A-MDL. The opposite direction holds iff $log_2\left(\frac{f_p}{440}\right) \in \mathbb{Z}$ & $t \in \mathbb{Q}$.*

**<u>Proof</u>**

Similar proof follows once we break down the polyphonic music into multi-channelled monophonic melodies with a lot of 'dummy' rests in the format of $\binom{N_{mod\ 12}}{S_I} \neq \binom{0}{-\infty}$ for S-MDL and $f_p \neq 0$ for A-MDL.

■

From Lemma 1 and Lemma 2, the following theorem holds:

*Theorem 1. A-S Theorem*

*S-MDL is a subset of A-MDL in terms of music descriptivity.*

**Proof**

From the proof of Lemma 1 and 2, every S-MDL can be converted into A-MDL, which means every vector of S-MDL can be transferred into a vector of A-MDL. On the other hand, for the opposite direction, we require the vector of A-MDL to satisfy the property: $\log_2\left(\frac{\mathrm{f_p}}{440}\right) \in \mathbb{Z}$. However, the range of $\log_2\left(\frac{\mathrm{f_p}}{440}\right)$ is $\mathbb{R}$ for $f_p \in \{0\} \cup \mathbb{R}^+$. Thus, the condition $\log_2\left(\frac{\mathrm{f_p}}{440}\right) \in \mathbb{Z} \subseteq \mathbb{R}$ indicates that S-MDL is a subset of A-MDL. Similarly for $\mathrm{t} \in \mathbb{Q} \subseteq \mathbb{R}$.

■

Based on S-MDL and A-MDL, here is the general definition of MDL.

*Definition 7. Music Definition Language (MDL).*

*Music Definition Language is a collection of vector sequences which represents the basic stream flow of the melody. It contains the fundamental frequency, the volume and the beginning time of the sound in either A-MDL or S-MDL form.*

### 4.2. Music Manipulation Language

Once we have described the basic sound features, we need to consider some extra information about the individual sound. Thus, as described in Chapter 3, this ‘language’ considers the extra information about how the individual notes should be played. In music notations, this ‘language’ is related to the articulation marks applied on the music note. From the reality, different artists might treat the same note differently. Especially, the live performance version might be different to album recording version. Moreover, this made the symbolic music closer to the audio music in terms of sound fidelity. Once again, similar to the concept of DML (Elmasri & Navathe, 2014), we have defined this ‘language’ as Music Manipulation Language (MML), literally meaning ‘manipulate the music’.

As stated in Section 2.5.4, the homeomophism concept from Topology has been used in our MML design, such that it can further group the similar treatments. Therefore, there are two (major) types of MML for our coding scheme.

The first type of MML is Operational Music Manipulation Language (O-MML) and the second one is Topological Music Manipulation Language (T-MML).

*Definition 8. Operational Music Manipulation Language (O-MML).*

*Operational Music Manipulation Language is a 'meaningful' function, either in matrix, vector or the combination form, such that it can be applied to those consecutive MDL vectors to perform linear transformations, even the A-MDL & S-MDL conversion.*

*Definition 9. Topological Music Manipulation Language (T-MML).*

*Topological Music Manipulation Language is a sequence of vectors, such that it can describe how, topologically, the notes were played under the defined MDL interval.*

From the two definitions, O-MML is aiming to identify the manipulation under similar linear transformations, for example, an entire song speeded up or slowed down. Adjusting the volume would not affect the main flow of the music. On the other hand, T-MML is aiming to identify similar music manipulation techniques applied on the single notes, such as various articulation marks.

The rest of Chapter 4 is concentrated on the T-MML. Before that, we briefly give some examples and ideas about the O-MML.

*Lemma 3. The C Key Transformation Lemma*

*There exists O-MML which allows music in any key to change into the C major (or A minor) key.*

**<u>Proof</u>**

From Equation (11-14), we can see that the conversion is respect to the 440 Hz where the note number is 0 and similarly, from Figure 2.5-5, A minor corresponds to C major. Then wherever we start counting, we just add or minus the number of extra keys between the new starting key and that of A. Similarly, if we define C as the base key, then, to transform music to a different key, we just need to shift the base key. For example, an S-MDL was in the G Key, which means that every time we see $\begin{pmatrix} N_{mod\ 12} \\ S_I \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \end{pmatrix}$, it is in fact $\begin{pmatrix} 3 \\ 0 \end{pmatrix}$. In other words, the base key has been shifted by the vector $\begin{pmatrix} -7 \\ 0 \end{pmatrix}$.

$$S\text{-}MDL_{M\ in\ C} = S\text{-}MDL_{M\ in\ G}. + \begin{pmatrix} -7 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \left( S\text{-}MDL_{N\ in\ G} + \begin{pmatrix} -7 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right)_i, \qquad \forall i \in \mathbb{N}^+$$

As $N_{mod\ 12} \in \{0, 1, \dots, 11\}$ by definition, we just need to do the following mapping:

$$\begin{pmatrix} -a \\ b \end{pmatrix} \to \begin{pmatrix} -a + 12 \\ b - 1 \end{pmatrix}, \qquad a \in \mathbb{N}^+ \;\&\; b \in \mathbb{Z}$$

(17)

Thus, in this case, we name $\begin{pmatrix} -7 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}_{mod\ 12}$ as the O-MML, as this resembles the conversion as Equation (17).

As a summary, $\begin{pmatrix} \pm k \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}_{mod\ 12}$, $k \in \{0, \dots, 11\}$ is the required O-MML which transfers the music in any key back to the C key, which ensembles the following conversion mapping:

$$\begin{cases} \begin{pmatrix} N_{mod\ 12} \\ S_I \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \to \begin{pmatrix} a - 12 \\ b + 1 \end{pmatrix}, & 13 \le a \le 23 \;\&\; b \in \mathbb{Z} \\ \begin{pmatrix} N_{mod\ 12} \\ S_I \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \to \begin{pmatrix} a + 12 \\ b - 1 \end{pmatrix}, & -11 \le a \le -1 \;\&\; b \in \mathbb{Z} \end{cases}$$

(18)

■

Similarly, we have the Set Transformation Lemma, Volume Tuning Lemma, Tempo Tuning Lemma, Extended A-MDL and S-MDL Conversion, and others. However, we have omitted the proof.

*Lemma 4. The Set Transformation Lemma*

*There exists O-MML which allows the music to change tone while staying in the same key.*

*Lemma 5. The Volume Tuning Lemma*

*There exists O-MML which allows the music to change the dynamic/volume.*

*Lemma 6. The Tempo Tuning Lemma*

*There exists O-MML which allows the music to change the tempo.*

*Lemma 7. The Extended A-MDL and S-MDL Conversion Lemma*

*Given the standard amplitude, tempo measurement and the time signature, there exists non-linear O-MML such that every S-MDL can be converted into A-MDL. The opposite direction holds iff* $log_2\left(\frac{f_p}{440}\right) \in \mathbb{Z}\ \&\ t \in \mathbb{Q}$.

As these O-MMLs, from the definition, can be applied on the whole music or partial melodies. Therefore, similar or repeated patterns inside a music can be identified easily. Moreover, these would be benefit our music applications. For example, we can alter the key in a similar way to many karaoke systems.

Going back to T-MML, every MDL should follow at least one T-MML, to record the actual 'manipulation' inside that MDL interval. Hence, we have the following features for T-MML with respect to A-MDL:

- The continuous change in frequency ($\Delta F$) inside the duration time of one single A-MDL vector. Thus, notes like Glissando, Turning notes, Arpeggiated chords, and even the continuous Glissando from EM can be coded with higher fidelity.
- The continuous change in amplitude ($\Delta A$) inside the duration time of one single A-MDL vector. Thus, notes like Accented notes and most musical dynamics can be encoded.
- The continuous change in time ($\Delta t$), allows users to record the time of any short note or break inside the duration time of one single A-MDL. Thus, notes like Staccato notes, and riffed notes from EM can be coded.

Similar T-MML features can be obtained for S-MDL. Thus, it can be summarized into the following equations, where '$\Delta$' indicates the difference between the start and the end of the MDL intervals:

$$\begin{cases} T\text{-}MML_{A-MDL} := \left( \begin{pmatrix} \Delta f_p \\ \Delta A \\ \Delta t \end{pmatrix} \right)_j \\ T\text{-}MML_{S-MDL} := \left( \begin{pmatrix} \Delta N_{mod\ 12} \\ \Delta S_I \\ \Delta A_d \\ \Delta B \\ \Delta B_T \\ \Delta D \end{pmatrix} \right)_j \end{cases}, \qquad \forall j \geq 1 \tag{19}$$

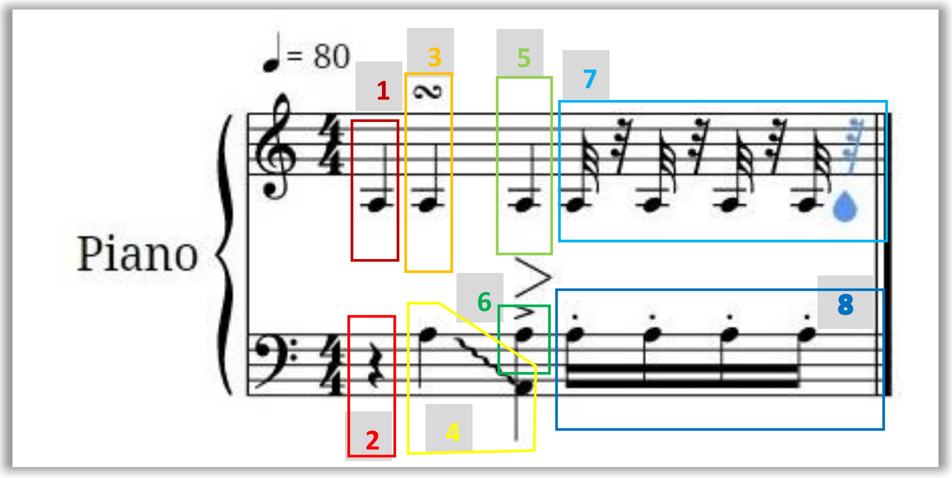
Figure 4.2-1 has shown the examples of the different uses of T-MML.

From Figure 4.2-1, Examples 1 and 2 have shown that the normal notes and the musical rests are coded differently in A-MDL/S-MDL, where they have the same T-MML. From the proof of Lemma 1, we allow the minimum capture pitch to be 6.875 $Hz$, as the first MIDI note is around 8.18 $Hz$.

From Examples 3 and 4, we can see the advantage of using T-MML to clarify the difference between the main melody flow and the extras from the turning note and the glissando note example. Moreover, S-MDL with T-MML is used for the traditional glissando note whereas M-MDL with T-MML is used for the continuous glissando note from EM.

From Example 5 and 6, we can see that T-MML is capable of representing the volume change and the accent note. This exemplifies the second feature of A-MDL ($A$) and T-MML ($\Delta A$).

Finally, Examples 7 and 8 illustrate two similar sounded melodies with different meanings and representations using musical notations. Example 7 is a demisemiquaver note with a demisemiquaver rest whereas example 8 is a staccato semiquaver note. From Figure 4.2-1, without T-MML, they have to have the same MDL (A-MDL or S-MDL). However, with T-MML, these two situations are no longer distinguishable.



Key:

S-MDL & T-MML

$$\begin{pmatrix} N_{mod\ 12} \\ S_I \\ A_d \\ B \\ B_T \\ D \end{pmatrix} \& \begin{pmatrix} \Delta N_{mod\ 12} \\ \Delta S_I \\ \Delta A_d \\ \Delta B \\ \Delta B_T \\ \Delta D \end{pmatrix}$$

A-MDL & T-MML

$$\begin{pmatrix} f_p \\ A \\ t \end{pmatrix} \& \begin{pmatrix} \Delta f_p \\ \Delta A \\ \Delta t \end{pmatrix}$$

| | S-MDL without T-MML | S-MDL/A-MDL with T-MML |
|---|---|---|
| 1 | $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \& \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}$ *or* $\begin{pmatrix} 220 \\ 10 \\ 0 \end{pmatrix} \& \begin{pmatrix} 0 \\ 0 \\ 0.75 \end{pmatrix}$ |
| 2 | $\begin{pmatrix} 0 \\ -5 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$[1] | $\begin{pmatrix} 0 \\ -5 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$[1] $\& \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}$ *or* $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \& \begin{pmatrix} 0 \\ 0 \\ 0.75 \end{pmatrix}$ |
| 3 | $\begin{pmatrix} 2 & 0 & 10 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1.25 & 1.5 & 1.75 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$[2] | $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \& \begin{pmatrix} 2 & -2 & -2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -0.25 & -0.25 & -0.25 & -0.25 \end{pmatrix}$[2] *or* $\begin{pmatrix} 220 \\ 10 \\ 0.75 \end{pmatrix} \& \begin{pmatrix} 26.942 & -26.942 & -24.002 & 24.002 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1875 & 0.1875 & 0.1875 & 0.1875 \end{pmatrix}$[2] |
| 4 | $\begin{pmatrix} 0 & 10 & 8 & 7 & 5 & 3 & 2 & 0 \\ 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 8/7 & 9/7 & 10/7 & 11/7 & 12/7 & 13/7 & 2 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1 \end{pmatrix}$[2] | $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 2 \end{pmatrix} \& \begin{pmatrix} 0 & 0 \\ -1 & 0 \\ 0 & -0.5 \\ 0 & 0 \\ 1 & 1 \\ -1 & -1 \end{pmatrix}$[2] *for discrete glissando* *or* $\begin{pmatrix} 220 \\ 10 \\ 0.75 \end{pmatrix} \& \begin{pmatrix} -110 & 0 \\ 0 & -5 \\ 0.75 & 0.75 \end{pmatrix}$[2] *for continuous glissando* |
| 5 | $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \end{pmatrix} \& \begin{pmatrix} 0 \\ 0 \\ -0.5 \\ 0 \\ 1 \\ -1 \end{pmatrix}$ *or* $\begin{pmatrix} 220 \\ 10 \\ 1.5 \end{pmatrix} \& \begin{pmatrix} 0 \\ -5 \\ 0.75 \end{pmatrix}$ |
| 6 | $\begin{pmatrix} 0 \\ 0 \\ 1.5 \\ 1 \\ 2 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 0 \\ 0 \\ 1.5 \\ 1 \\ 2 \\ 1 \end{pmatrix} \& \begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 1 \\ -1 \end{pmatrix}$ *or* $\begin{pmatrix} 220 \\ 15 \\ 1.5 \end{pmatrix} \& \begin{pmatrix} 0 \\ -10 \\ 0.75 \end{pmatrix}$ |
| 7 | $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -5 & 0 & -5 & 0 & -5 & 0 & -5 \\ 0.5 & 0 & 0.5 & 0 & 0.5 & 0 & 0.5 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 3 & 3.125 & 3.25 & 3.375 & 3.5 & 3.625 & 3.75 & 3.875 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \end{pmatrix}$[1][2] | $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -5 & 0 & -5 & 0 & -5 & 0 & -5 \\ 0.5 & 0 & 0.5 & 0 & 0.5 & 0 & 0.5 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 3 & 3.125 & 3.25 & 3.375 & 3.5 & 3.625 & 3.75 & 3.875 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \end{pmatrix}$[1][2] $\& \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.125 \\ -0.125 \end{pmatrix}$ *or* $\begin{pmatrix} 220 & 0 & 220 & 0 & 220 & 0 & 220 & 0 \\ 5 & 0 & 5 & 0 & 5 & 0 & 5 & 0 \\ 2.25 & \frac{75}{32} & \frac{78}{32} & \frac{81}{32} & \frac{84}{32} & \frac{87}{32} & \frac{90}{32} & \frac{93}{32} \end{pmatrix}$[2] $\& \begin{pmatrix} 0 \\ 0 \\ \frac{3}{32} \end{pmatrix}$ |
| 8 | $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -5 & 0 & -5 & 0 & -5 & 0 & -5 \\ 0.5 & 0 & 0.5 & 0 & 0.5 & 0 & 0.5 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 3 & 3.125 & 3.25 & 3.375 & 3.5 & 3.625 & 3.75 & 3.875 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \end{pmatrix}$[1][2] | $\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 1 & 1 & 1 & 1 \\ 3 & 3.25 & 3.5 & 3.75 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$[2] $\& \begin{pmatrix} 0 \\ 0 \\ -0.5 \\ 0 \\ 0.25 \\ -0.125 \end{pmatrix}$ *or* $\begin{pmatrix} 220 & 220 & 220 & 220 \\ 5 & 5 & 5 & 5 \\ 2.25 & \frac{78}{32} & \frac{84}{32} & \frac{90}{32} \end{pmatrix}$[2] $\& \begin{pmatrix} 0 \\ -5 \\ \frac{3}{32} \end{pmatrix}$ |

*Figure 4.2-1. & Table 4.2-1. Examples of MDL and T-MML with music notations.*

[1] Assume the minimum capture pitch is 6.875 Hz. By default and definition, this represents the rest note in S-MDL form.

[2] To save space, we grouped the S-MDL, A-MDL & T-MML vector sequences into one matrix respectively.

Cut along this line

Fold along this line

Make sure this page is evenly numbered.

From these examples, we can conclude that the T-MML is necessary to be added onto the MDL in representing the music, where MDL corresponds to our existing symbolic coding schemes, such that both the fidelity is closer to the actual audio and the coding is closer to the actual music sheet. For example, if we want to have a higher fidelity for a non-linear continuous glissando note, we can just add more T-MMLs to a A-MDL to get a better linear approximation. However, for certain musical content analysis, it does not matter whether it is a linear approximation or non-linear original glissando note, as long as they are homeomorphic, i.e., $p_{lin.approx}(t = 0) = p_{non-lin.ori}(t = 0)$ & $p_{lin.approx}(t = 1) = p_{non-lin.ori}(t = 1)$ where $p$ is the pitch function against time, $t$, $t = 0$ is the start of the A-MDL and $t = 1$ is the end of the A-MDL (ref: Section 2.5.4).

Based on O-MML and T-MML, we have derived the following definition for MML.

*Definition 10. Music Manipulation Language (MML).*

*Music Manipulation Language is used to describe how different artists performed the MDL in more technical details, either in the collection of topological vector form or as a mapping function.*

### 4.3. Grammar and Rules

For the completion of the language setting, this section briefly introduce some typical grammar or rules for MDL and MML.

Since the main concepts of MDL and MML were come from DDL and DML respectively, we can use similar instructions as SQL to further modify the MDL and MML sequences while we transfer from theories to applications. Here are some theoretical examples:

- To create a new MDL sequence, we may use the syntax '*create* ... *as*'. E.g., *Create* (Sequence_number) *as* S-MDL or *Create* (Sequence_number) *as* A-MDL.
- To insert any instrumental setting, we may use the syntax '*ins-set*'. E.g., *Create* (Sequence_number) *as* S-MDL *ins-set* (Instrument). Similar syntax can be defined for other settings, for example, tempo setting would be '*tem-set*'.
- To adjust any existing settings, simply replace '*create*' with '*modify*' and '*set*' with '*adj*'. E.g., *Modify ins-adj* (Instrument)
- To remove an existing MDL sequence, we may use the syntax '*remove*'. E.g., *Remove* S-MDL (Sequence_number) (Sequence_number)

- To insert or delete any individual MDL, we may use the syntax '*insert … to … at …*' or '*delete … from … of …*'. E.g., *Insert* S-MDL (S-MDL parameters) *to* (Sequence_number) *at* (Position_number). Similarly, we can insert or delete any individual T-MML.
- To adjust any existing MDL from the sequence, we may use the syntax '*adjust*'. E.g., *Adjust* S-MDL (new S-MDL parameters) *for* (Sequence_number) (Position_number). Similarly, we can adjust any existing T-MML.
- To define O-MML, we may use the syntax '*def-omml*'. E.g., *def-omml* (O-MML parameters).

The syntax described here can be adjusted if any confusion is found. Moreover, we can separate MDL (A-MDL or S-MDL) and T-MML by forcing all numerical entries in T-MML to be signed. E.g., Example 1 in Figure 4.2-1, $\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0.75} \end{pmatrix}$ becomes $\begin{pmatrix} \mathbf{+0} \\ \mathbf{+0} \\ \mathbf{+0.75} \end{pmatrix}$. Therefore, we can run a systematic check to validate whether the MDL or MML is inserted individually.

## 4.4. Music Representation using the MDL and MML

With MDL and (T-)MML, a proof-of-concept prototype has been built that illustrates the way of generating music or soundwaves using the defined coding scheme, in order to show some features claimed previously about music representation (Dannenberg, 1993).

Refering to the case study in Section 2.2, we use this typical example, the introduction part of "Zero For Conduct" by Bastarz (of Block B) [45], to illustrate the potential of MDL and MML, as at this stage we manually implemented MDL and MML. Figure 4.4-1 has shown the Spectral Pitch Display (SPD) for the official music track using Adobe Audition.

From Figure 4.4-1, we can see that the introduction can be split into two channels. The first one is the audio channel that covers the continuous change in the frequency ($f \in \mathbb{R}$) and the 'real' time ($t \in \mathbb{R}$) that is hard for the existing symbolic approach to cover. The second one is the symbolic channel that involves polyphonic music, multiple sound channels and sound resources, which is able to be covered by the existing audio and symbolic approaches.

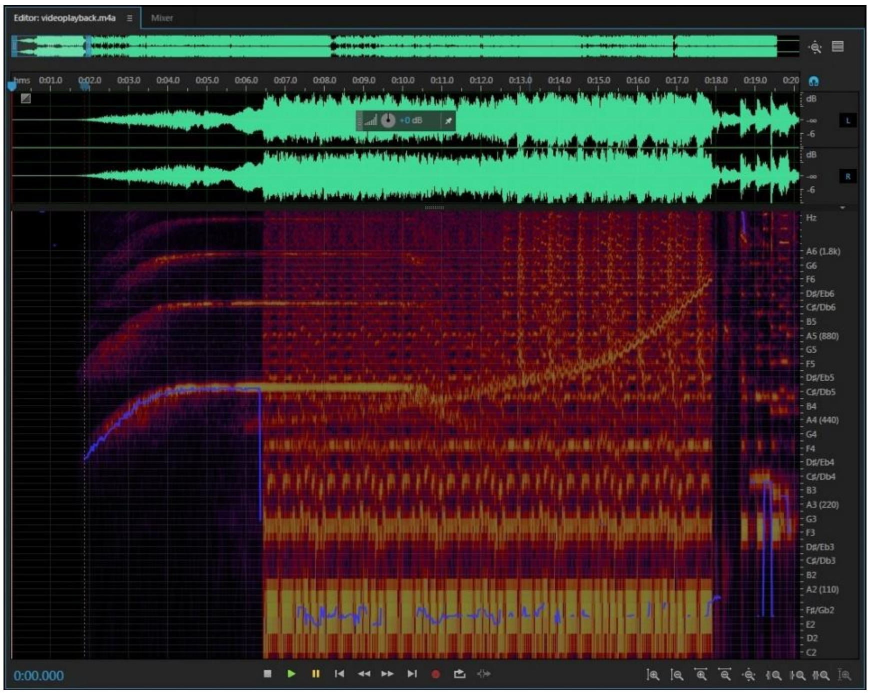


*Figure 4.4-1. Spectral Pitch Display of the introduction of [45] using Adobe Audition.*

From Lemma 2 and Definition 10, we can see that both MDL and MML can extend to polyphonic music. Thus, better sound quality can be achieved by ensembling the harmony channel. Therefore, for simplicity, we demonstrate the SPD with fundamental frequencies only, using the manually implemented MDL and T-MML for the intro of [45]. For the same reason, every MDL follows one T-MML for every note. Thus, several 'dummy' MDL and one extra channel for the chord have been manually added. More specifically, for the audio channel, we have used the A-MDL structure and the corresponding T-MML. On the other hand, for the symbolic channel, we have also used the A-MDL structure and the last three entries of S-MDL as bonus features for separation, and the corresponding T-MML.

I.e., $\begin{pmatrix} f_p & \Delta f_p \\ A & \Delta A \\ t & \Delta t \end{pmatrix}$ & $\begin{pmatrix} f_p & \Delta f_p \\ A & \Delta A \\ t & \Delta t \\ B & \Delta B \\ B_T & \Delta B_T \\ D & \Delta D \end{pmatrix}$ respectively.

The manual MDL and MML were stored in Excel format of the file size of 15 KB, as it would be easy for the MatLab to load. Therefore, Algorithm 1 shows the procedure of reading the MDL and MML file and generating the SPD.

*Algorithm 1. MDL&MML Playback (simplified SPD version)*

**Input: MDL and MML (Excel Format)**

**Output: Spectral Pitch Display**

```
1. Read MDL and T-MML and relevant parameter initializations.
2. Split into corresponding channels.
3. $resolution\left(\frac{1}{r}\right) := time\ per\ beat\ \left(\frac{60}{x}\right) * min.duration\ \left(\frac{l}{\beta_{min}}\right)$.
4. $r_t(i), i \in \left\{0, \left(\frac{1}{r(t_{m.end}-t_{m.begin})}\right) \ldots, 1\right\}$ where $r_t(0) = t_{m.begin}, r_t(1) = t_{m.end}$.
5. For each channels (audio or symbolic)
   a. Calculate the A-MDL frame (the number of notes or the number of MDL and MML pair).
   b. For each $A\text{-}MDL$ frame
      i. $t_{min} = A\text{-}MDL(3);\ t_{max} = A\text{-}MDL(3) + T\text{-}\mathrm{MML}(3)$
      ii. $s_t(j), j \in \left\{0, \left(\frac{1}{r(t_{max}-t_{min})}\right) \ldots, 1\right\}$ where $s_t(0) = t_{min}, s_t(1) = t_{max}$, $s_t(j) \subseteq r_t(i)$ and $\bigoplus_{\forall\ A\text{-}MDL\ frame} s_t = r_t$.
      iii. $G_{A,s_t} = \frac{T\text{-}MML(2)}{T\text{-}MML(3)};\ I_{A,s_t} = A\text{-}MDL(2) - A\text{-}MDL(3) * G_{A,s_t}$
      iv. $A(s_t) = G_{A,s_t} \times s_t + I_{A,s_t}$
      v. Rescale into dBs ($-1$ dB $\approx$ 0.891 Amp. & $-\infty$ dB = 0 Amp.).
      vi. $G_{f,s_t} = \frac{T\text{-}MML(1)}{T\text{-}MML(3)};\ I_{f,s_t} = A\text{-}MDL(1) - A\text{-}MDL(3) * G_{f,s_t}$
      vii. $f(s_t) = G_{f,s_t} \times s_t + I_{f,s_t}$
      viii. Convert $f(s_t)$ to $n_p$ using Equation (12), regardless of $n_p \in \mathbb{Z}$.
   c. End
6. End
7. Concatenate $n_p$ & $A(s_t)$ across all the channels.
8. Plot SPD in real-time, including Amplitude-time graph ($A(s_t)\ in\ dBs$ against $t$), Pitch-time graph ($n_p$ against $t$).
```

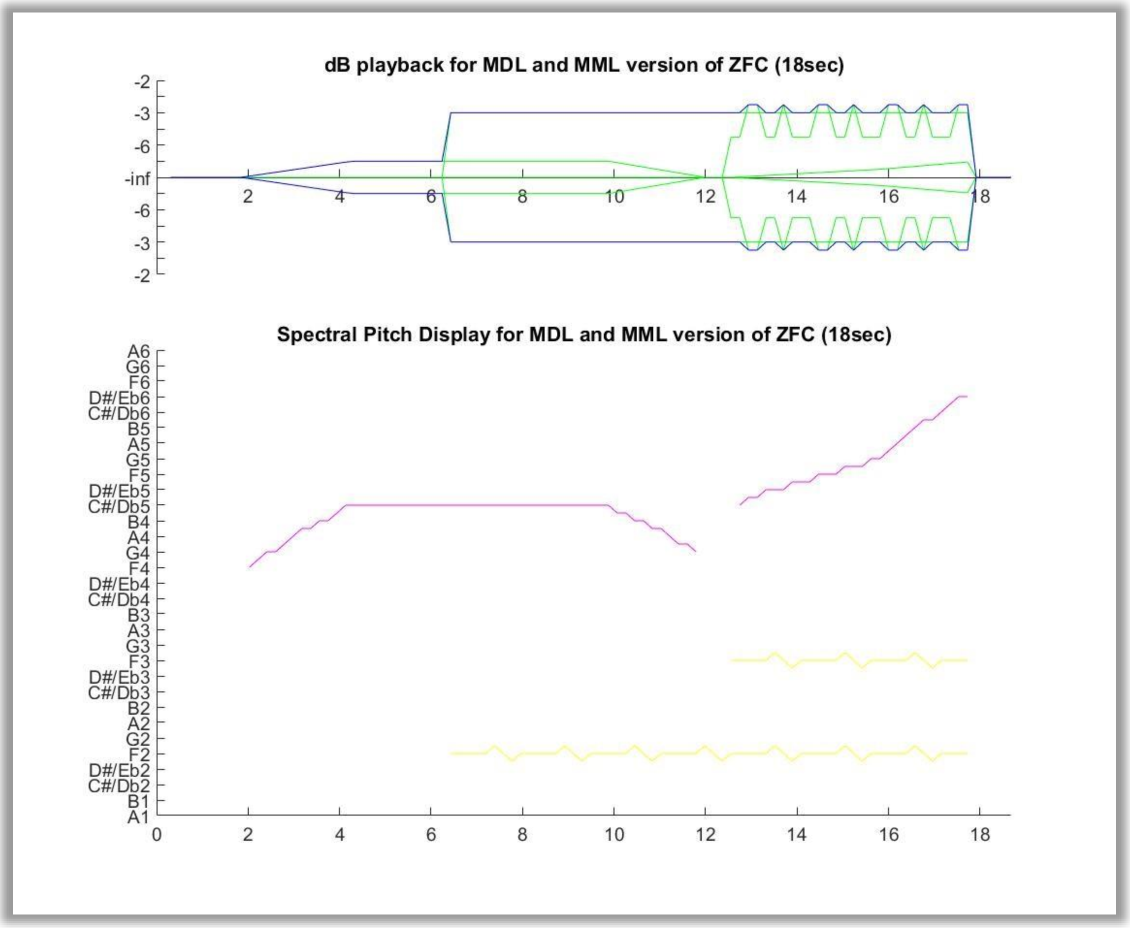


*Figure 4.4-2. Simplified SPD of the introduction of music track [45] using MatLab.*

Based on the Algorithm 1, the following simplified SPD has been generated, as shown in Figure 4.4-2.

In Figure 4.4-2, the green line is the amplitude for the individual channel whereas the blue line is the concatenated amplitude. The magenta line is the fundamental frequency, in terms of $n_p$, generated from the audio channel whereas the yellow line is generated from the symbolic channel. More specifically, the magenta line represents the continuous glissando using the linear line.

By comparing Figure 4.4-1 and Figure 4.4-2 with Figure 4.4-1 and Figure 4.4-3, we can see that the main melody flow of the music almost matches in terms of fundamental pitch, amplitude, and time, as those lines fit the original music's SPD much better than using the MIDI cover's SPD against the original SPD. In Figure 4.4-3, only the symbolic part has been transferred into the piano cover (MIDI version).

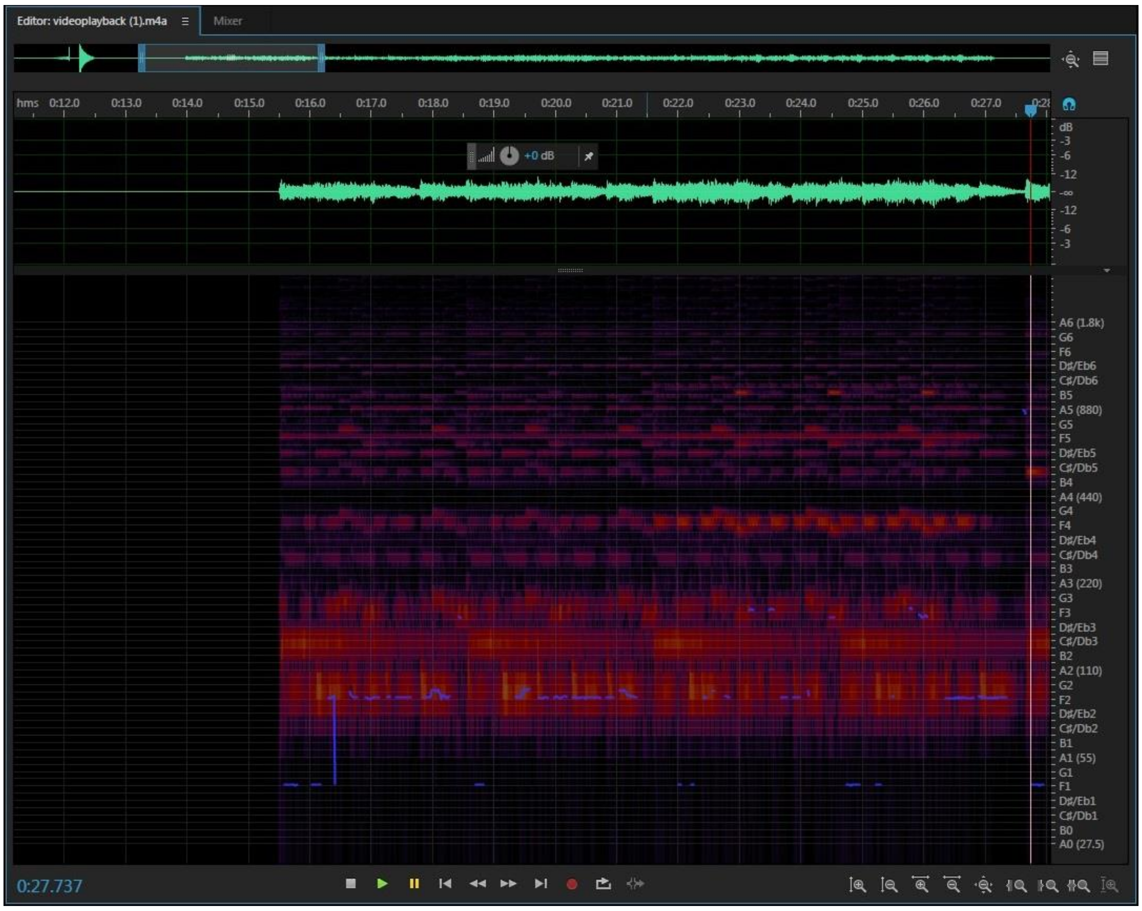


*Figure 4.4-3. Spectral Pitch Display of the introduction of [45a] using Adobe Audition.*

From a file storage perspective, our file is only 15 KB for the first 18 seconds that covers almost all of the sound, apart from the harmonies, whereas the whole song ([45]) is 3.01 MB in ‘M4A’ format, 2.96 MB in ‘MP3’ format and the short (symbolic) MIDI cover ([45a]) is 6.53 KB. Thus, with appropriate estimation, we can claim that the file size of the MDL and MML file lays in between of the audio and the symbolic files as expected.

On the other hand, the MIDI cover is not included the audio channel. Hence, from the music representation, we can claim that MDL and MML is an improvement of existing symbolic files and towards the audio files. In fact, even if we try to ensemble the audio channel into the MIDI cover, it would be a number of step functions, rather than a linear function.

Therefore, based on this slightly complicated case, the MDL and MML file is claimed to lay in between audio and symbolic files in terms of music storage and representation.

However, unlike SPD, music is a real-time event and the soundwave is generated as time goes on. Despite Algorithm 1 providing a real-time SPD, we still need to see the actual synthezised accoustic sound of the MDL and MML.

As this is not the main concept of the project, Xue (2018) has helped complete the simplified relevant work as his final year project . He obtained a result of that, with human listeners and cosidering the normal version, the version with continuous glissando and the version with chord for the music track [27] by applying Equation (20) using Python, MDL and MML has correctly represented the melody flow. However, the sound quality is not particularly high quality as if only considers the fundamental frequencies (Xue, 2018).

$$\omega(t) = \left(A + \frac{\Delta A}{\Delta t} t_s\right) \sin\left(2\pi\left(f_p + \frac{\Delta f_p}{\Delta t} t_s\right) t\right) \tag{20}$$

where $t_s \in \{0, \dots, 1\}$, linear spacing the range [0,1] for sound sampling within one MDL.

Therefore, in the section, we have shown that by using MDL and MML to store the music data, it is more capacity-efficient in terms of file size and has better sound retrieval where sound waveforms are concerned. More importantly, it can handle a greater variety of audio signals compared to the existing MIDI format. However, the sound quality possesses a lot of room for improvement.

## 4.5. Convert Audio into the MDL and MML using Signal Processing

In order to show the one-to-one mapping from the coding and the accoustic sound (Fremerey, et al., 2008), we now need to prove the converse, from music to the code. I.e., convert into MDL and MML from the audio like MIDI (Derrien, 2014). For a similar reason, we build a proof-of-concept prototype for a simple case.

The input is the MP3 file implemented in Section 4.4, the normal version of music track [27], i.e., $\Delta f_p = \Delta A = 0$, as we generate the sound using the manually noted S-MDL and T-MML file, in Excel format. The recording has a sample rate of 44.1 kHz and consists of two channels with equal sinusoidal values, which is similar to recording without a microphone or downloading online music. The output is the similarity score between the

*Algorithm 2. MDL & T-MML Generation (simplified, assuming $\Delta f_p = \Delta A = 0$)*

**Input: MP3 File**

**Output: A-MDL/(S-)MDL & T-MML Files**

```
1. Read MP3 file and parameter initializations (e.g., $\theta$).
2. Display basic MP3 information.
3. Generate the sequence of the sinusoidal value, $\omega(t_i)$.
4. Plot the time-domain figure ($\omega(t_i)$ against $t$).
5. $silent_{left}(k) = t_i$, where $\omega(t_{i-1}) \neq 0$ & $\omega(t_i) = \omega(t_{i+1}) = 0, \forall k$.
6. $silent_{right}(k) = t_j$, where $\omega(t_{j-1}) = \omega(t_j) = 0$ & $\omega(t_{j+1}) \neq 0, \forall k$.
7. $l_s(k) = silent_{right}(k) - silent_{left}(k), \forall k$.
8. Filter $l_s(k)$ into $l_s(k')$ s.t. $l_s(k') \geq \theta$.
9. $number\ of\ note = k'$.
10. For each note (m = 1 to $k'$)
    a. Apply FFT on the sinusoidal value. % Discrete sound waves (Section 2.5.3)
    b. Extract $f_p(m)$ & $A(m)$.
    c. Transfer $l_s(k')$ into time in seconds, $\Delta t(m)$.
    d. $t(m) = \sum_{n=1}^{k'-1} \Delta t(n)$
    e. Plot all the relevant graphs and display the four features.
    f. $\begin{cases} \text{A-MDL}\ (\text{m}, 1) = f_p(m) \\ \text{A-MDL}\ (\text{m}, 2) = A(m) \\ \text{A-MDL}\ (\text{m}, 3) = t(m) \\ \text{T-MML}_A\ (\text{m}, 1) = 0 \\ \text{T-MML}_A\ (\text{m}, 2) = 0 \\ \text{T-MML}_A\ (\text{m}, 3) = \Delta t(m) \end{cases}$, based on Equation (10) and (19)
    g. Convert into (S-)MDL(m) and the corresponding T-MML_S(m) using 'similar' steps from Lemma 1, i.e., Equation (21).
11. End
12. Concatenate A-MDL (m)/(S-)MDL (m) & T-MML (m) , for all m.
13. Output A-MDL/(S-)MDL & T-MML Files as Excel format.
```

***Algorithm 3. Similarity Evaluation Between Manually and Automatic Generated (S-)MDL&MML Files***

**Input: Manually and Automatic Generated (S-)MDL & T-MML File from Algorithm 2.**

**Output: Similarity Scores, $\text{Sim}_C$ & $\text{Sim}_R$ or $\text{Sim}_{length}$.**

```
1.  Read both files and any parameter initializations (e.g., $\vartheta$).
2.  Extract both melody and rhythm line:
    $\text{m}_1(\text{t}) = \text{(S-)MDL}_{man}(1)$ & $\text{r}_1(\text{t}) = \text{(S-)MDL}_{man}(5)$
    $\text{m}_2(\text{t}) = \text{(S-)MDL}_{auto}(1)$ & $\text{r}_2(\text{t}) = \text{(S-)MDL}_{auto}(5)$[1]
3.  If $\text{length}(\text{m}_1(\text{t})) == \text{length}(\text{m}_2(\text{t}))$, similar for $\text{r}_1(\text{t})$ and $\text{r}_2(\text{t})$
    a.  $l = \text{length}(\text{m}_1(\text{t}))$
4.  Else
    a.  $\text{Sim}_{length} = \frac{\text{length}(\text{m}_2(\text{t})) - \text{length}(\text{m}_1(\text{t})),}{\text{length}(\text{m}_1(\text{t})),}$ [2]
    b.  Output $\text{Sim}_{length}$
5.  End If
6.  For $\text{m}_1(\text{t})$ & $\text{m}_2(\text{t})$ % with Equation (22)
    a.  Count the number of allowed captured pitches against $\vartheta$, $|A|$.
    b.  Count the number of disallowed captured pitches against $\vartheta$, $|B|$.
    c.  Count the number of perfectly captured pitches, $|C|$.
7.  End
8.  For $\text{r}_1(\text{t})$ & $\text{r}_2(\text{t})$ % with Equation (24)
    a.  Count the number of perfectly matched duration length, $|R|$.
9.  End
10. $\text{Sim}_C = \frac{|A|}{l} * e^{-MSD} = \frac{|A|}{l} * e^{-(|A|+|B|-|C|)^{-1} * \sum_{t=1}^{l} (\text{m}_2(\text{t}) - \text{m}_1(\text{t}))^2}$, Equation (23)
11. $\text{Sim}_R = \frac{|R|}{l}$, % with Equation (24)
12. Output $\text{Sim}_C$ & $\text{Sim}_R$.
```

[1]: Equation (21) will allow the rest of (S-)MDL & T-MML elements to match *iff* these two lines match.

[2]: At a later stage, will call the normalization function to make sure the comparison melody lengths were the same.

manually noted S-MDL and T-MML file against the file converted from the audio using signal processing. The whole process is split into Algorithm 2 and Algorithm 3.

As any noise will affect the FFT process, the fundamental pitch will not be identical to the actual pitch. Hence, in order to convert into S-MDL at Algorithm 2 line 10.g, we need to use the rounding function to make sure the note output $\begin{pmatrix} N_{mod\ 12} \\ S_I \end{pmatrix}$ is, by definition, in integer. Therefore, to measure the error of the full process, frequency is stored in the (S-)MDL file instead. I.e., (S-)MDL & T-MML$_S$(m) in Algorithm 2 & Algorithm 3 is in the format of $\begin{pmatrix} f_p & \Delta f_p \\ A_d & \Delta A_d \\ B & \Delta B \\ B_T & \Delta B_T \\ D & \Delta D \end{pmatrix}$, where $\Delta \mathrm{f_p} = \Delta \mathrm{A}_d = 0$.

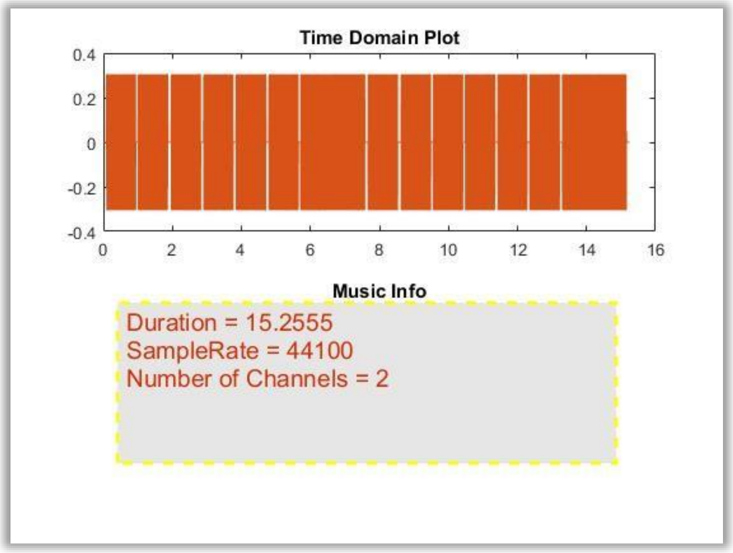
On the other hand, the recording volume will affect the actual amplitude in the time-domain figure, whereas the duration of the note, $\Delta t(m)$, can be irrational.

Thus, Algorithm 2 line 10.g considered the following instead:

$$\begin{cases} \text{(S-)MDL(m, 1)} := f_p = f_p(m) \in \mathbb{R} \\ \text{(S-)MDL(m, 2)} := A_d = round(\dfrac{A(m)}{A(1)}) \in \mathbb{Q}^+ \\ \text{(S-)MDL(m, 3)} := B = \dfrac{\sum_{n=1}^{m-1} D(n) - B_T(m)}{\beta} + 1 \in \mathbb{N} \\ \text{(S-)MDL(m, 4)} := B_T = \displaystyle\sum_{n=1}^{m-1} D(n)\ mod\ \beta \in \mathbb{Q}^+ \qquad , \qquad \forall \mathrm{m} \in \{1, \dots, k'\} \\ \text{(S-)MDL(m, 5)} := D(m) = round\left(\dfrac{t(m)}{t(1)}\right) \in \mathbb{Q}^+, w.r.t. \dfrac{1}{\beta}_{min} \\ \text{T-MML}_S\text{(m, 3)} := \Delta B(m) = 0 \\ \text{T-MML}_S\text{(m, 4)} := \Delta B_T(m) = D(m) \\ \text{T-MML}_S\text{(m, 5)} := \Delta D(m) = -D(m) \end{cases} \tag{21}$$

where letting the first note's amplitude is 1 and defining the duration of the first note ($t(1)$) is equal to 1 as it starts with a crotchet note in the time signature of $\frac{4}{4}$, $\beta = 4$.

In Algorithm 3, the following equations were considered:



Duration = 15.2555
SampleRate = 44100
Number of Channels = 2

*Figure 4.5-1. Main time domain plot and the music information for music track [27].*

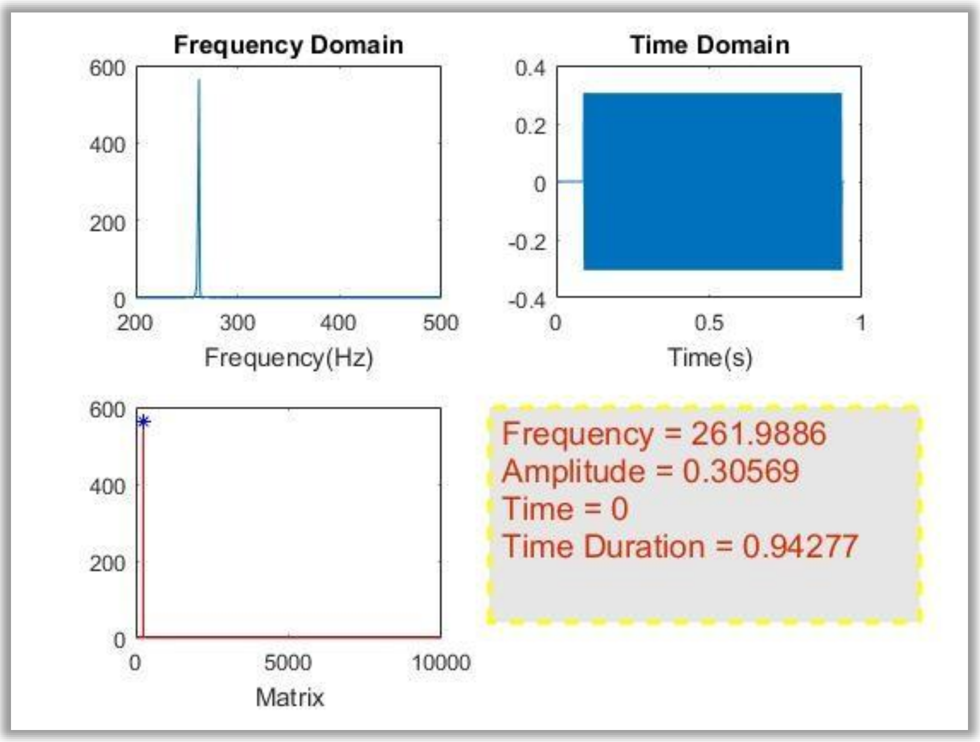
$$\begin{cases} A = \{i \big| |m_1(i) - m_2(i)| \le \vartheta,\ i \in \{1, \dots, l\}\} \\ B = \{i||m_1(i) - m_2(i)| > \vartheta, i \in \{1, \dots, l\}\} \\ C = \{i \big| |m_1(i) - m_2(i)| = 0, i \in \{1, \dots, l\}\} \end{cases} \quad \text{(line 6)} \tag{22}$$

$$MSD = \frac{1}{|A| + |B| - |C|} \sum_{t=1}^{l} \left(m_2(t) - m_1(t)\right)^2 \quad \text{(line 10)} \tag{23}$$

where MSD is known as the mean squared distance (without square root).

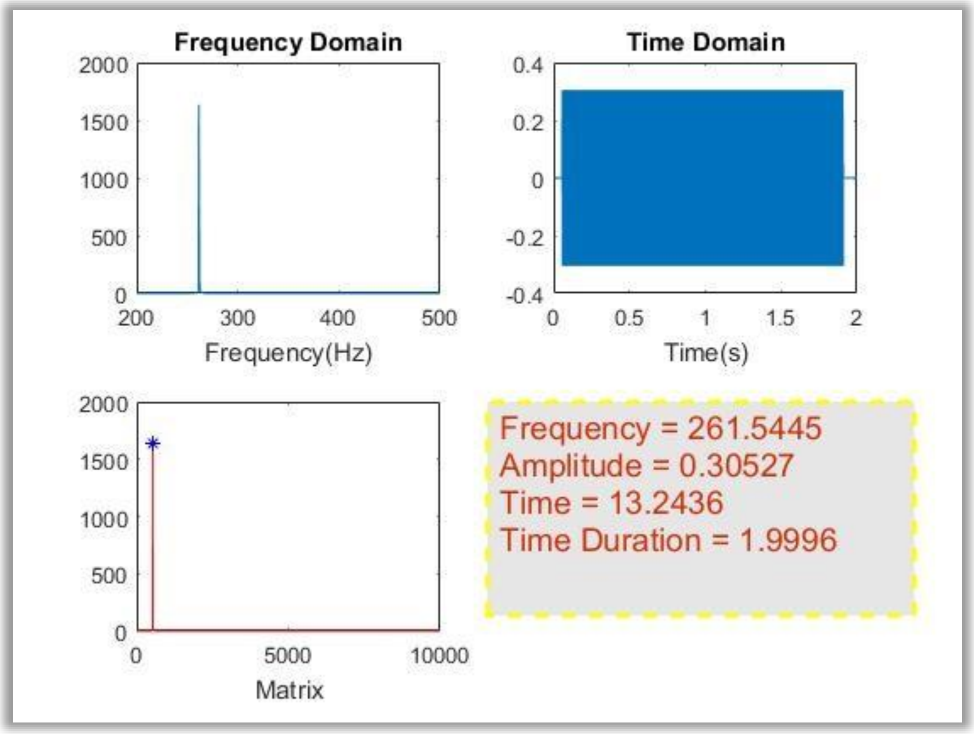
$$R = \{i||r_1(i) - r_2(i)| = 0\} \quad \text{(line 8.a, 11)} \tag{24}$$

After running the proof-of-concept prototype with the sample track [27], the following results have been obtained. Figure 4.5-1 is the main plot for the whole MP3 file at Algorithm 2 line 4. Figure 4.5-2 and Figure 4.5-3 are the sample plots for the first and the last note, containing the frequency domain figure, time domain figure and the four features extracted at Algorithm 2 line 10.e. Figure 4.5-4 shows the output of the A-MDL and T-MML file in Excel format. Figure 4.5-5 shows the output of the (S-)MDL and T-MML file.



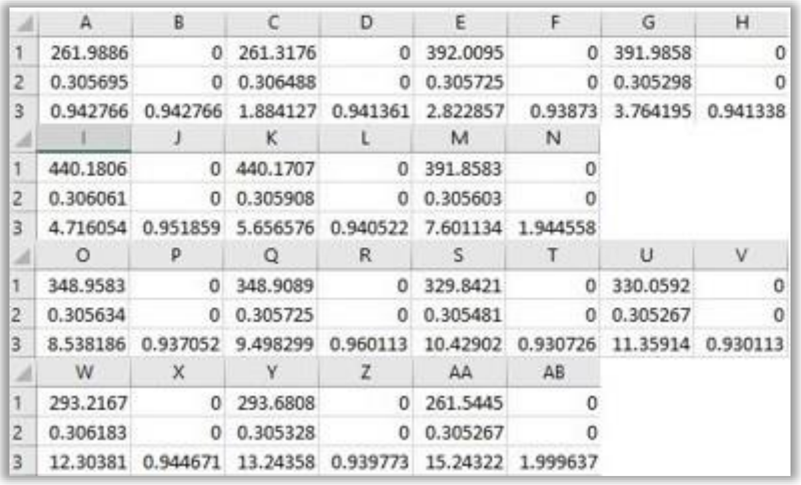
Frequency = 261.9886
Amplitude = 0.30569
Time = 0
Time Duration = 0.94277

*Figure 4.5-2. Plots and feature output for the first note of music track [27].*



Frequency = 261.5445
Amplitude = 0.30527
Time = 13.2436
Time Duration = 1.9996

*Figure 4.5-3. Plots and feature output for the last note of music track [27].*



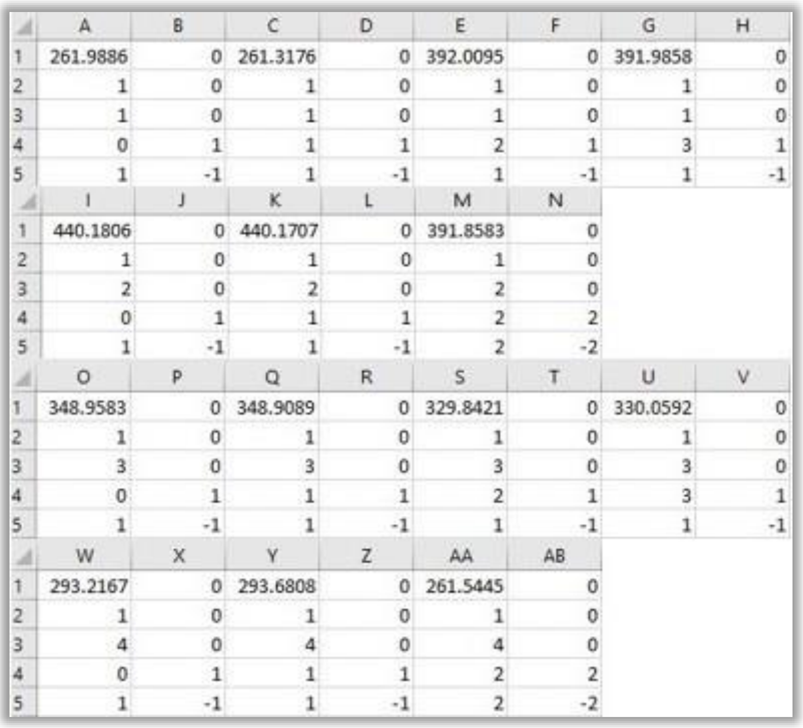
| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | 261.9886 | 0 | 261.3176 | 0 | 392.0095 | 0 | 391.9858 | 0 |
| 2 | 0.305695 | 0 | 0.306488 | 0 | 0.305725 | 0 | 0.305298 | 0 |
| 3 | 0.942766 | 0.942766 | 1.884127 | 0.941361 | 2.822857 | 0.93873 | 3.764195 | 0.941338 |

| | I | J | K | L | M | N |
|---|---|---|---|---|---|---|
| 1 | 440.1806 | 0 | 440.1707 | 0 | 391.8583 | 0 |
| 2 | 0.306061 | 0 | 0.305908 | 0 | 0.305603 | 0 |
| 3 | 4.716054 | 0.951859 | 5.656576 | 0.940522 | 7.601134 | 1.944558 |

| | O | P | Q | R | S | T | U | V |
|---|---|---|---|---|---|---|---|---|
| 1 | 348.9583 | 0 | 348.9089 | 0 | 329.8421 | 0 | 330.0592 | 0 |
| 2 | 0.305634 | 0 | 0.305725 | 0 | 0.305481 | 0 | 0.305267 | 0 |
| 3 | 8.538186 | 0.937052 | 9.498299 | 0.960113 | 10.42902 | 0.930726 | 11.35914 | 0.930113 |

| | W | X | Y | Z | AA | AB |
|---|---|---|---|---|---|---|
| 1 | 293.2167 | 0 | 293.6808 | 0 | 261.5445 | 0 |
| 2 | 0.306183 | 0 | 0.305328 | 0 | 0.305267 | 0 |
| 3 | 12.30381 | 0.944671 | 13.24358 | 0.939773 | 15.24322 | 1.999637 |

*Figure 4.5-4. A-MDL and T-MML output for music track [27].*



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | 261.9886 | 0 | 261.3176 | 0 | 392.0095 | 0 | 391.9858 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 1 | 2 | 1 | 3 | 1 |
| 5 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 |

| | I | J | K | L | M | N |
|---|---|---|---|---|---|---|
| 1 | 440.1806 | 0 | 440.1707 | 0 | 391.8583 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 2 | 0 | 2 | 0 | 2 | 0 |
| 4 | 0 | 1 | 1 | 1 | 2 | 2 |
| 5 | 1 | -1 | 1 | -1 | 2 | -2 |

| | O | P | Q | R | S | T | U | V |
|---|---|---|---|---|---|---|---|---|
| 1 | 348.9583 | 0 | 348.9089 | 0 | 329.8421 | 0 | 330.0592 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 3 | 0 | 3 | 0 | 3 | 0 | 3 | 0 |
| 4 | 0 | 1 | 1 | 1 | 2 | 1 | 3 | 1 |
| 5 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 |

| | W | X | Y | Z | AA | AB |
|---|---|---|---|---|---|---|
| 1 | 293.2167 | 0 | 293.6808 | 0 | 261.5445 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 4 | 0 | 4 | 0 | 4 | 0 |
| 4 | 0 | 1 | 1 | 1 | 2 | 2 |
| 5 | 1 | -1 | 1 | -1 | 2 | -2 |

*Figure 4.5-5. (S-)MDL and T-MML output for music track [27].*

Let $\vartheta = 0.5\,Hz\,(approx. 8.66 - 8.18)$, then the (S-)MDL and T-MML file and the initial file gave 93.63% accuracy for the contour melody and 100% accuracy for the rhythm. More specifically, it proves the following features of MDL and MML that was described at the design stage:

Firstly, if we allow small errors, then everything would be 100% accurate as we will round to the nearest integer when converting into the proper S-MDL and T-MML file. This means the 6.37% error is all due to the background noise when recording, as all the frequencies extracted are within 0.5 $Hz$.

Secondly, we can see that each crotchet lasts about 0.95 seconds. This means that we can change the time setting to make the music play faster or slower without affecting the fundamental pitch value using O-MML.

Thirdly, if the first note is not the standard duration, crotchet in our example, we can use O-MML to make the modification. For example, if the first note is a quaver instead, then mulitiply all duration by 0.5 and you can obtain the corrected duration value. Thus, the bar number and the beat number can be modified.

The above two points would be useful when proving Lemma 6. The Tempo Tuning Lemma.

Finally, we have made the similar assumption that the amplitude of the first note is 1, whereas the real amplitude is approximately 0.305, which may be due to the volume reduction when recording. The amplitudes of all the rest of the notes are the relative values compared to the first note. For a similar reason, by defining the first note to a certain dynamic value, the volume for the entire music can be controlled by an O-MML. Similarly, this helps to prove Lemma 5. The Volume Tuning Lemma.

## 4.6. Summary

We have introduced the MDL and MML with various examples, including the potential of using MDL and MML to model the music with various articulation marks. We have outlined several mathematical theories relevant to the new coding scheme. Most importantly, we have illustrated the one-to-one mapping between the acoustic sound and the coding scheme and showed the advantages of using the MDL and MML with proof-of-concept prototypes, from both music storage and music representation perspectives.

# 5. Original Music Identifier using MDL and MML

Given the good result previously, the one to one mapping between the music and MDL&MML, we now move towards the main MUCASM for OMI. We, first design the similarity measurement and the mechanism for MUCASM, followed by various experimental studies to see how the ensembled learning can improve the OMI's accuracy. Moreover, it will further prove that MDL and MML lays in between the audio and the traditional symbolic coding schemes.

## 5.1. MUCASM Design

According to prototype and the test requirements in Chapter 3, we want to see the difference between just using unsupervised SOM and the supervised RL-based ensembled learning. Thus, with the defined MDL and the the corresponding T-MML, we design our MUsic Classification And Similarity Mesurement (MUCASM) system.

From Chapter 4, we are assuming that MDL & T-MML can be well used to represent and store music melodies. Thus, the input melody to MUCASM is encoded with the proposed scheme MDL, which includes S-MDL and A-MDL with the corresponding T-MML. Then we compare the input with those in the original music (melody) database following both audio and symbolic based similarity measurement methods to identify matched ones based on given criteria (i.e. the highest similarity score). After sorting the outputs, the system retrieves the most matched original music (melody) track IDs and relevant information. The output can either be unique, multiple or none depending on the criteria. Moreover, the melody database does not only store original melodies, but also their variations, in order to provide a rich data set to train the proposed classification algorithms. Figure 5.1-1 shows the overall MUCASM architecture.

Apart from MML and MDL, the similarity calculation and classification (i.e. self-supervised reinforcement-based ensemble learning for the musical origin classification) also play pivotal roles in the proposed system. The first subsection describes various similarity calculation methods based on SOM (Self-Organize Map), and the second subsection explains the proposed classification approaches based on Ensemble Machine Learning (EML) which includes several learning mechanisms and these are various SOMs.

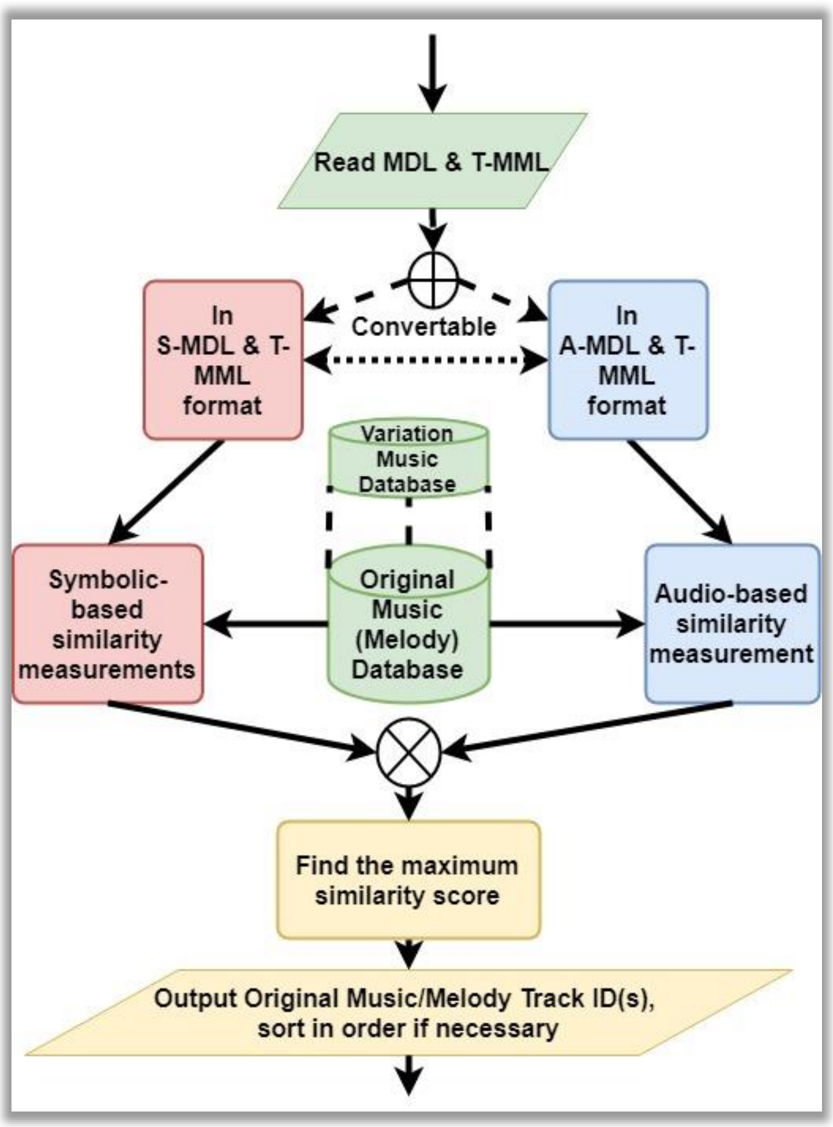

Read MDL & T-MML
In S-MDL & T-MML format
Convertable
In A-MDL & T-MML format
Variation Music Database
Symbolic-based similarity measurements
Original Music (Melody) Database
Audio-based similarity measurement
Find the maximum similarity score
Output Original Music/Melody Track ID(s), sort in order if necessary

*Figure 5.1-1. Overall flowchart for MUCASM.*

In Figure 5.1-1, $\oplus$ is the 'Or' symbol, which indicates that the process flow continues in more than two branches; $\otimes$ is the 'Summing Junction' Symbol, which indicates a point in the flowchart where multiple branches converge back into a single process*.

### 5.1.1. Self-Organizing Map

SOM, a self-learning method, can produce different effectiveness on clustering according to the selection of embedded similarity calculations, activation functions for the best match unit and the update formulae. In order to maximise the SOM performance, and find their correlation and optimum combination, the activation function is fixed, but the methods for similarity calculation and update formula are varied.

* https://www.smartdraw.com/flowchart/flowchart-symbols.htm

#### 5.1.1.1. Similarity Calculations

For S-MDL-based SOM, the two important features extracted from the sextuple S-MDL are the contour melody and the rhythm. The contour melody differs from the Parsons code for melodic contours (Müllensiefen & Wiggins, 2011), and is the sequence of the key number derived from $\binom{N_{mod\ 12}}{S_I}$ against the duration $(D)$. The rhythm part is purely derived from $(D)$. Theoretically, $A_d, B, B_T$ should be taken into account at the pre-processing stage. However, as they were either identical in our dataset or can be derived from the duration, we have excluded them at this time. Therefore, we have designed the following three equations for the contour melody and one for the rhythm given that the two melodies have the same length.

Similarity Calculation 1 for Contour Melody (SC1):

$$Sim_C\left(MDL_{1,i}(j), MDL_{2,i}(j)\right) = \frac{|A|}{6 * l}\ where\ A = \{(i,j)\,\big|\,|MDL_{1,i}(j) - MDL_{2,i}(j)| = 0\} \tag{25}$$

Similarity Calculation 2 for Contour Melody (SC2):

$$Sim_C = \frac{|A|}{6 * l} * e^{-MSD}\ where \begin{cases} A = \{(i,j)\,\big|\,|MDL_{1,i}(j) - MDL_{2,i}(j)| = 0\} \\ B = \{(i,j)\,\big|\,|MDL_{1,i}(j) - MDL_{2,i}(j)| \neq 0\} \\ MSD = \frac{1}{|B|} * \sum_{i,j \in B} \left(MDL_{1,i}(j) - MDL_{2,i}(j)\right)^2 \end{cases} \tag{26}$$

Similarity Calculation 3 for Contour Melody (SC3):

$$Sim_C = w_M * Sim_M + w_D * Sim_D + b\ where \begin{cases} Mel_{1,i} = MDL_{1,i}(1) + 12 * MDL_{1,i}(2) \\ Mel_{2,i} = MDL_{2,i}(1) + 12 * MDL_{2,i}(2) \\ A = \{i\,\big|\,|MDL_{1,i}(6) - MDL_{2,i}(6)| = 0\} \\ B = \{i\,\big|\,|Mel_{1,i} - Mel_{2,i}| \neq 0\} \\ MSD = \frac{1}{|B|} * \sum_{i \in B} \left(Mel_{1,i} - Mel_{2,i}\right)^2 \\ Sim_M = e^{-MSD} \\ Sim_D = \frac{|A|}{l} \end{cases} \tag{27}$$

Rhythm Coding (RC):

$$\begin{cases} 1 & there\ is\ a\ note\ pressed \\ 0 & otherwise \end{cases},$$
$$\text{in the resolution of } \frac{1}{s} \in \frac{1}{2^n} \text{and the time signature of } \frac{\alpha}{\beta}. \tag{28}$$

Similarity Calculation for Rhythm (SR):

$$Sim_R\left(MDL_{1,i}(j), MDL_{2,i}(j)\right) = \frac{|A|}{l}\ where \begin{cases} Rhy_{1,i} \overset{(28)}{\longleftarrow} MDL_{1,i}(6) \\ Rhy_{2,i} \overset{(28)}{\longleftarrow} MDL_{2,i}(6) \\ A = \{i\,|\,|Rhy_{1,i} - Rhy_{2,i}| = 0\} \end{cases} \tag{29}$$

SC1 is simply the ratio between the number of identical entries and the total number of entries. In Equation (25), $i$ indicates that it is the ith MDL in the sequence, $j$ is the column index of the vector-based MDL and $l$ is the length of the $MDL_{1,i}(j)$, or $MDL_{2,i}(j)$, as they have normalized into the same length. However, this method has a limitation that it cannot distinguish between a melody with small variance and a melody with big variance, as they might give the same similarity output. For example, one original melody consists of five notes, but Variance 1 and Variance 2 both have one note different. Despite the musical note distance between the original melody and Variance 1 is closer than with Variance 2, SC1 will give both case 80% similarity scores.

Thus, SC2 made an improvement. It used Mean Squared Distance (MSD) to capture the possible detailed difference. As this is primarily for symbolic MDL, thus unlike the equation in Algorithm 3 line 10, only two sets were defined, either the entries from two melodies were identical, set A, or different, set B. This means, despite the fact that we might have the same fraction $\frac{|A|}{l}$, as we multiply a different factor of $e^{-MSD}$, the final similarity score will vary. Notice that, in Equation (26), $i, j\ and\ l$ have the same meaning.

For SC3, we made another improvement. As the musical melody is frequency against time, we split the contour melodic similarity scores into two parts, using the idea of perceptron neural network (Widrow & Lehr, 1990; Heaton, 2015). One for the similarity in terms of melody and one in terms of the duration such that the overall similarity is the sum of those two similarities with two weights, $w_M$ *and* $w_D$, where $w_M + w_D = 1$, and a bias, $b$, where we set it to be 0. For the melodic similarity, we only use the exponential of the negative MSD to capture the detailed difference. Thus, we need to convert into the melody lines, $Mel_{1,i}$ *and* $Mel_{2,i}$, from the first two entries of S-MDL, and define set B. For the rhythmic similarity, we just use the simple fraction as there is no detailed difference. Thus, set A is defined using $j = 6$. Notice that, in Equation (27), $i$ *and* $l$ have the same meaning.

For all those three equations, firstly, $MDL()$ was meant to be $S\text{-}MDL_{Melody}$ at this stage. Secondly, they are for comparing two vector sequences of the equal length. Therefore, most of the time, we need Equation (30) to normalize two melodies with unequal length.

$$\begin{cases} \widetilde{MDL}_{1,\tilde{\imath}_i \dots \tilde{\imath}_{i+1}-1}(j) \leftarrow MDL_{1,i}(j) \;\; for\; j = 1, \dots, 4 \\ \widetilde{MDL}_{1,\tilde{\imath}_i \dots \tilde{\imath}_{i+1}-1}(5) \leftarrow MDL_{1,i}(5) + \frac{\beta}{s} * dist\left(MDL_{1,i}(5)\right)_{\frac{1}{s}} \\ \widetilde{MDL}_{1,\tilde{\imath}_i \dots \tilde{\imath}_{i+1}-1}(6) \leftarrow MDL_{1,i}(6) - \frac{\beta}{s} * dist\left(MDL_{1,i}(5)\right)_{\frac{1}{s}} \\ \widetilde{MDL}_{2,\tilde{\imath}_i \dots \tilde{\imath}_{i+1}-1}(j) \leftarrow MDL_{2,i}(j) \;\; for\; j = 1, \dots, 4 \\ \widetilde{MDL}_{2,\tilde{\imath}_i \dots \tilde{\imath}_{i+1}-1}(5) \leftarrow MDL_{2,i}(5) + \frac{\beta}{s} * dist\left(MDL_{2,i}(5)\right)_{\frac{1}{s}} \\ \widetilde{MDL}_{2,\tilde{\imath}_i \dots \tilde{\imath}_{i+1}-1}(6) \leftarrow MDL_{2,i}(6) - \frac{\beta}{s} * dist\left(MDL_{2,i}(5)\right)_{\frac{1}{s}} \end{cases},$$

$$\text{in the resolution of } \frac{1}{s} \in \frac{1}{2^n} \text{ and the time signature of } \frac{\alpha}{\beta}. \tag{30}$$

where $\tilde{\imath}$ is the index where the whole melody duration with time signature of $\frac{\alpha}{\beta}$ has evenly distributed in the resolution of $\frac{1}{s}$. $\tilde{\imath} = \left\{1, \dots, \frac{s}{\beta} * \alpha * number\ of\ bars\right\}$, where $\frac{1}{s}$ must be less than the minimum duration of two melodies. For example, four bars with time signature, $\frac{4}{4}$, in the resolution of $\frac{1}{16}$, $\tilde{\imath} = \{1, .., 256\}$. In this example, neither of the two melodies should contain a duration less than $\frac{1}{16}$. However, it is possible that melody A's minimum duration is $\frac{1}{16}$ and melody B's minimum duration is $\frac{1}{8}$.

Moreover, $dist(MDL)_{\frac{1}{s}}$ is the evenly distributed set of the duration of the MDL under the resolution $\frac{1}{s}$, and $D = 1$ if the note is crotchet when $\beta = 4$. From the same example above, if the value of $MDL_{1,i=1}(6) = 1$, then $dist\left(MDL_{1,i}(6)\right)_{\frac{1}{s}} = \{0,1,2,3\}$ as $\frac{s}{\beta} = 4$. Thus, $\widetilde{MDL}_{1,\tilde{\imath}=1}(6) = 1 + 0.25 * 0 = 1$, $\widetilde{MDL}_{1,\tilde{\imath}=2}(6) = 1 + 0.25 * 1 = 1.25$ and so on until when $\widetilde{MDL}_{1,\tilde{\imath}=5}(6) = MDL_{1,i=2}(6)$. This means we can create an index subsequence $\tilde{\imath}_i$ such that $\tilde{\imath}_i \in \tilde{\imath}$ and $i$ is the index of the original MDL. Thus, from the same example above, $\tilde{\imath}_1 = 1; \tilde{\imath}_2 = 5$ etc.

Similarly, for RC, the rhythm pattern is in the resolution of $\frac{1}{s}$. Therefore, after we have extended the $MDL$ into $\widetilde{MDL}$, we code 1 in every position $\tilde{\imath}_i$, which indicates that this is the time that the note has been pressed, and 0 otherwise, as these were the 'dummy' notes. For the same example, the rhythm will become $(1,0,0,0,1,\dots)$ using Equation (28).

As the rhythm pattern is either 0 or 1, Mean Squared Distance is no longer needed. Hence, Equation (29) is sufficient to evaluate the similarity score.

Finally, Equation (31) evaluates the final similarity scores:

$$Sim\left(MDL_{1,i}(j), MDL_{2,i}(j)\right) = w_C * Sim_C + w_R * Sim_R + b \tag{31}$$

where $Sim_C$ use either SC1, SC2 or SC3, and $b$ has set to be zero.

##### 5.1.1.2. Self-learning

The first approach when updating the weight at the self-training stage is to follow the traditional SOM self-learning (Section 2.5.1.2). This means we treat the MDL sequence as 'weights' and the melody itself as hidden neuron. Thus, Equation (32) has been considered:

Self-Learning 1 (SL1):

$$T_n(i,j,t+1) = T_n(i,j,t) - \eta\left(T_n(i,j,t) - C(i,j,t)\right) \tag{32}$$

where $T_n()$ represents the value of the selected $n^{th}$ hidden neuron, $i$ is the index of the MDL melody sequence, $j$ is the column index of the individual MDL, $t$ is the iteration time, $\eta$ is the learning rate and $C()$ is the current input.

However, as the duration is normally in the form of $\frac{a}{2^n}$, for some integer $a$ and $n$, it would be sensible to update the relevant field in a similar way, such that it would benefit for the rhythmic similarity calculation. Equation (33) allows the relevant value of the hidden neuron approach to value of the selected input much faster, thus the similarity will make a significant change. Otherwise, as an example, 0.95 would still not match with the target input 0.5 after one iteration, where $\eta = 0.1$ and the original value of the hidden value is 1.

$$\begin{cases} T_n(i,j,t+1) = T_n(i,j,t) - \eta\big(T_n(i,j,t) - C(i,j,t)\big), \qquad j = 1,2,3,4 \\ T_n(i,5,t+1) = T_n(i,5,t) + \frac{\beta}{s} * \operatorname{sgn}\Big(\eta\big(T_n(i,5,t) - C(i,5,t)\big)\Big) \\ T_n(i,6,t+1) = T_n(i,6,t) - \frac{\beta}{s} * \operatorname{sgn}\Big(\eta\big(T_n(i,6,t) - C(i,6,t)\big)\Big) \end{cases},$$
$$where\ \frac{1}{s} \in \frac{1}{2^n} \text{ is the resolution and } \frac{\alpha}{\beta} \text{ is the time signature.} \tag{33}$$

Now, for the same example above, the hidden value 1 will become 0.5 after two iterations, for $\eta = 0.1$, the resolution is $\frac{1}{16}$ and the time signature is $\frac{\alpha}{\beta} = \frac{4}{4}$.

*Algorithm 4. SOM for S-MDL*

**Input:** S-MDL self-training sets, $S\text{-}MDL_{{Melody}_{ori,var}}$.

**Output:** Accuracy for the self-testing, $acc$

```
1. Read $S\text{-}MDL_{{Melody}_{ori,var}}$ and parameter initializations (e.g., learning rate, $\eta$ and number of iterations $k$).
2. Define hidden neurons from $S\text{-}MDL_{{Melody}_{ori,var}}$, $h_p$, where $p$ is the original melody's index.
       $\mathcal{H} = \{h_p,\ \forall p \in \mathbb{N}\} \subseteq \mathcal{M} = \{S\text{-}MDL_{{Melody}_{ori,var}},\ \forall ori, var \in \mathbb{N}\}$
% For Self-training.
3. For iteration = 1 to k
    a. For each input from $S\text{-}MDL_{{Melody}_{ori=x,var=y}}$.
        i. Normalization over all $h_p$ using Equation (30) respectively.
        ii. Evaluate Similarity score over all $h_p$ using Equation (31) (Equation (26)/Equation (27) only when $w_R = 0$).
        iii. $Sim_{max} = Sim\left(S\text{-}MDL_{{Melody}_{x,y}}, h_{p_{max}}\right) = max\left\{Sim\left(S\text{-}MDL_{{Melody}_{x,y}}, h_p\right),\ \forall p \in \mathbb{N}\right\}$ (as the activation function)
        iv. Return index $p_{max}$.
        v. Update $h_{p_{max}}$ using Equation (32) or Equation (33) where $T_n = h_{p_{max}}, n = p_{max}\ and\ C = S\text{-}MDL_{{Melody}_{x,y}}$, and the learning rate $\eta$.
    b. End for
4. End for
% For Self-testing.
5. Repeat step 3.a.i upto 3.a.iv for all $S\text{-}MDL_{{Melody}_{ori,var}}$ in the training sets to obtain a confusion matrix.
6. $acc = \frac{|\{(ori,var),\ p_{max}=ori\}|}{|ori|*|var|}$, from the confusion matrix.
7. Output $acc$
```

##### *5.1.1.3. Algorithm for SOM*

Based on the idea and mechanism of SOM and those similarity calculations and learning methods, we have constructed Algorithm 4. The input is the melody training set in S-MDL format, ${S\text{-}MDL_{Melody}}_{ori,var}$ using Equation (8), where $ori$ is index of the original track and $var$ is the index for the variational type. The initial hidden neurons were set to be those original melodies, where the index for the varitional type is 0. The output is the index number, $i$, of most likely to be an original melody from variational melodies after clustering.

#### 5.1.2. RL-based EML

At this stage, we need to maintiain the advantages of audio-based MIRs, an audio approach for A-MDL and T-MML is implemented, i.e., audio fingerprinting. With all the components, we outline the process flow for our EML approach on MUCASM for OMI.

##### *5.1.2.1. Audio fingerprinting*

Similar to the existing audio fingerprinting, we, firstly, need to create an audio fingerprint hash table. Thus, after evaluating several existing audio-fingerprint approaches, we designed the following process flow such that the proposed hybrid system can use both symbolic (Algorithm 4) and audio approaches. The process flow considers A-MDL and T-MML. For simplicity, the amlitude and the frequency remain constant.

This process was inserted before the self-training stage, and contains a fingerprint extraction function in order to create a fingerprint database that corresponds to Shazam fingerprint catalog (Haitsma & Kalker, 2002; Haitsma & Kalker, 2003; Wang, 2003; Cano, et al., 2005; Duong & Duong, 2015). Therefore, similar to most audio fingerprinting methods, it involves the following stages:

- Pre-processing:
  We create different samples with size 50%, 200%, 500% or 1000% by extracting or interpolating the original data. For example, if the original data contains 10 MDL, we implement a sequence of 5, 20, 50 and 100 vectors respectively, which consider either the pitch and the duration for S-MDL or the Shazam-like fingerprint catalogue for A-MDL.

- Feature Extraction:

  We extract the distinct frequencies from previous vector sequences into our audio fingerprint hash table.
- Post-processing:

  We count the appearence time either using the tally count or using the actual duration time. However, only the original music has been stored at this stage, as we need different songs to have distinctive high peak pitches and less affected by the variations.
- Feature Modelling:

  The numbers of distinctive frequencies stored in our hash table is defined by the compression ratio, 50% or 100%. For example, a 50% compression ratio means we only the use top five frequencies and the scores in our final table, when we have found ten distinct frequencies. The final table were sorted in the appearance of time order, from high to low.

Once we have the hash table, we outlined the following equation for evaluating the similarity score between the two hash tables, the hash table for the current input, $T_C$, and, either the ith originals or the ith hidden neurons after SOM, $T_{O_i\ or\ H_i}$.

$$Sim_H = \frac{|A|}{|A| + |B|} + e^{-MSD}\ where \begin{cases} A = \left\{ i \middle| \left| T_C(i,1) - T_{O_i\ or\ H_i}(i,1) \right| = 0 \right\} \\ B = \left\{ i \middle| \left| T_C(i,1) - T_{O_i\ or\ H_i}(i,1) \right| \neq 0 \right\} \\ MSD = \frac{1}{|B|} * \sum_{i \in B} \left( T_C(i,1) - T_{O_i\ or\ H_i}(i,1) \right)^2 \end{cases} \tag{34}$$

For the same reason as Equation (25) – (27), we need to modify the two hash tables such that they have the same number of rows. Thus, we choose the comparison length that is the minimum of the hash tables. I.e. $|A| + |B| = \min_i \left\{ |T_C(i,1)|, \left| T_{O_i\ or\ H_i}(i,1) \right| \right\}$.

Thus, Algorithm 5 and Algorithm 6 have shown the process flow for the fingerprint extraction and the fingerprint matching for our MUCASM respectively, which relates to Figure 2.1-2.

*Algorithm 5. Audio Fingerprint Extraction*

**Input: S-MDL and corresponding T-MML, $S\text{-}MDL\&MML_{Melody}$.**

**Output: Audio Hash Table, $T$.**

```
1. Read $S\text{-}MDL\&MML_{Melody}$, based on Equation (8) & (19) and parameter
   initializations (E.g., the Sampling Rate, $R$, and the Compression Rate, $C$)
   % Pre-processing
2. $L = length(S\text{-}MDL\&MML_{Melody})$
3. For i = 1 to R*L
   a. Get the current frequency, $f_i$, and the current calculated duration
      time, $D_i$, based on the information from the corresponding T-MML.
      % Feature Extraction
   b. If $f_i \neq f_j, \forall j < i$
      i. Initialize $t(f_i) = 0$
      ii. $H(x,1) \leftarrow f_i\ , x = 1,2,3,\ldots respectively$. Add $f_i$ at the new $x^{th}$
          row of $H$.
          % Post-processing
      iii. $H(x,2) \leftarrow t(f_i) = 1\ OR\ H(x,2) \leftarrow t(f_i) = D_i, for\ the\ same\ x$.
   c. Else If $f_i = f_j, for\ some\ j < i$
      i. $H(x,2) \leftarrow t(f_i) = t(f_i) + 1\ OR\ H(x,2) \leftarrow t(f_i) = t(f_i) +$
         $D_i, for\ the\ corresponding\ x\ where\ H(x,1) = f_j$.
   d. End If
4. End For
5. $T \xleftarrow{\downarrow H(x,2)} H$, sort $H$ in order of $t(f_i)$, from high to low.
   % Feature Modelling
6. Given $C$, filter $T$.
```

*Algorithm 6. Audio Fingerprint Matching*

**Input: Hash tables generated by Algorithm 5, $T_C, T_{O_i}$ *and* $T_{H_i}, \forall i$.**

**Output: Similarity Scores, $Sim_H$ and the original melody index, $i$.**

```
1. Read Hash Tables and parameter initialization.
2. For i = 1 to n
    a. Evaluate $Sim_{H,i,1}$ for $T_C$ and $T_{O_i}$ using Equation (34).
    b. Evaluate $Sim_{H,i,2}$ for $T_C$ and $T_{H_i}$ using Equation (34).
3. End For
4. Find $\max_i Sim_{H,i,1}$ and $\max_i Sim_{H,i,2}$
5. If $\operatorname{argmax}_i Sim_{H,i,1} = \operatorname{argmax}_i Sim_{H,i,2}$
    a. $i = \operatorname{argmax}_i Sim_{H,i,1} = \operatorname{argmax}_i Sim_{H,i,2}$
    b. $Sim_H = \max_i Sim_{H,i,1}$
6. Else If $\operatorname{argmax}_i Sim_{H,i,1} \neq \operatorname{argmax}_i Sim_{H,i,2}$
    a. $i = \operatorname{argmax}_i Sim_{H,i,2}$ iff the confusion matrix improves in terms of the
       overall accuracy, while training.
    b. $i = \operatorname{argmax}_i(Sim_{H,i,1}, Sim_{H,i,2})$ & $Sim_H = \max_i(Sim_{H,i,1}, Sim_{H,i,2})$,
       otherwise.
7. End For
```

#### 5.1.2.2. *The Process Flow for EML*

As we now need both MDL and T-MML, Algorithm 4 has been modified with the following changes:

- The input is MDL (either S-MDL or A-MDL or the combination form in Section 4.5) and the corresponding T-MML.
- As T-MML is only involved when using the audio fingerprint, and combined with the symbolic-based similarity measurement, we may call Algorithm 6 when necessary.
- For A-MDL or the combination form, the melody equation in Equation (27) changes to $\begin{matrix} Mel_{1,i} = A\text{-}MDL_{1,i}(1) \\ Mel_{2,i} = A\text{-}MDL_{2,i}(1) \end{matrix}$', whereas for the combination form only, the duration index has changed to 5 instead of 6 (3 instead of 6 for A-MDL). We labelled as 'modified Equation (27)'. Similar modification and labelling applied to Equation (29) and Equation (30). Moreover, when evaluating symbolic-based similarity scores, only MDL is considered. In other words, we filtered out the T-MML. Therefore, nothing needs to be added for Equation (25) ~ (31).
- The following were added to Equation (33) when self-learning from the definition of $T\text{-}MML_{S\text{-}MDL}$, was labelled 'modified Equation (33)':

$$\begin{cases} T_n(i,j,t+1) = T_n(i,j,t) - \eta\big(T_n(i,j,t) - C(i,j,t)\big), \qquad j = 1,2,3,4 \\ T_n(i,5,t+1) = T_n(i,5,t) - \dfrac{\beta}{s} * \operatorname{sgn}\Big(\eta\big(T_n(i,5,t) - C(i,5,t)\big)\Big) \\ T_n(i,6,t+1) = T_n(i,6,t) + \dfrac{\beta}{s} * \operatorname{sgn}\Big(\eta\big(T_n(i,6,t) - C(i,6,t)\big)\Big) \end{cases} , for\ T\text{-}MML$$

Algorithm 7 shows the modified version of Algorithm 4, where $MDL\&T\text{-}MML_{Melody_{ori,var}}$ is defined as $(MDL_{Note}\quad T\text{-}MML_{MDL})_i$ over the index of the original melody and the index of the variation type, and *i* is the note index of the melody:

*Algorithm 7. SOM for MDL&MML*

```
Input: MDL&MML self-training sets, $MDL\&MML_{Melody_{ori,var}}$.

Output: Accuracy for the self-testing, $acc$

1. Read $MDL\&MML_{Melody_{ori,var}}$ and parameter initializations (e.g., learning
   rate, $\eta$ and number of iterations $k$).
2. Define hidden neurons from $MDL\&MML_{Melody_{ori,var}}$, $h_p$, where $p$ is the
   original melody's index.
   $\mathcal{H} = \{h_p,\ \forall p \in \mathbb{N}\} \subseteq \mathcal{M} = \{MDL\&MML_{Melody_{ori,var}},\ \forall ori, var \in \mathbb{N}\}$
% For Self-training.
3. For iteration = 1 to k
   a. For each input from $MDL\&MML_{Melody_{ori=x,var=y}}$.
      i. Normalization over all $h_p$ using Equation (30) respectively.
      ii. Evaluate Similarity score over all $h_p$ using Equation (31) (Equation
          (26)/modified Equation (27) only when $w_R = 0$) or Algorithm 6.
      iii. $Sim_{max} = Sim\left(MDL\&MML_{Melody_{x,y}}, h_{p_{max}}\right) =$
           $max\left\{Sim\left(MDL\&MML_{Melody_{x,y}}, h_p\right),\ \forall p \in \mathbb{N}\right\}$ (Activation func.)
      iv. Return index $p_{max}$.
      v. Update $h_{p_{max}}$ using Equation (32) or modified Equation (33) where
         $T_n = h_{p_{max}}, n = p_{max}\ and\ C = MDL\&MML_{Melody_{x,y}}$, and the
         learning rate $\eta$.
   b. End for
4. End for
% For Self-testing.
5. Repeat step 3.a.i upto 3.a.iv for all $MDL\&MML_{Melody_{ori,var}}$ in the training
   sets to obtain a confusion matrix.
6. $acc = \frac{|\{(ori,var),\ p_{max}=ori\}|}{|ori|*|var|}$, from the confusion matrix.
7. Output $acc$
```

From this modified SOM mechanism, a Self-supervised Reinforcement-based Ensemble Learning mechanism is proposed.

As each SOM with different combination of the equations may give different effects against different variation types, an Ensemble Learning mechanism is proposed, so we can merge the Audio-Fingerprinting-based approach described above and those Symbolic-based approaches. We treat each SOM with different similarity calculations and updating formulae as individual Tier 1 classifier, so we can form the final Tier 2 mega-classifier, where each Tier 1 classifiers has combined together to optimize the performance against each variation type. As we are currently dealing with small size data, we have removed the 'bootstrap' & 'jackknife' resampling stages. Thus, Figure 5.1-2 shows the 'Stacked Generalization' ensemble learning method used in our MUCASM.

Normally, music comes with certain labels, not just the music itself. This means we can use the labels, variation types in our case, to make unsupervised learning mechanism supervised. In other words, we can let the system learn which Tier 1 classifier (Algorithm 1 with different similarity calculations and update formulae) is the most suitable one for each variation type. Hence, during the training stage, we compare the performance against individual variation type after each epoch, by the accuracy of the confusion matrix. We may duplicate the Tier 1 classifiers from the previous epoch, as we cannot guarantee that the accuracy will improve after training for all the SOM algorithms. Therefore, we define those accuracies as rewards. After applying the 'Comparison between algorithms' from RL to those rewards, a decision tree, more specifically, a strategy tree, is generated. Hence, the mega-classifier will follow the strategy tree and optimize all the Tier 1 classifiers. Therefore, the RL-based learning mechanism allows the system to learn how the variations classify back to its origin, as well as which measurement is needed for different variation types, which has shown in Figure 5.1-3.

Figure 5.1-4 shows the mechanism for self-supervised learning. Self-supervised learning is a supervised learning where the data labels (e.g., variation type) were determined from the input data, not from environment.

Figure 5.1-5 shows the diagram of the SOM and the possible process flow the strategy tree may follow.

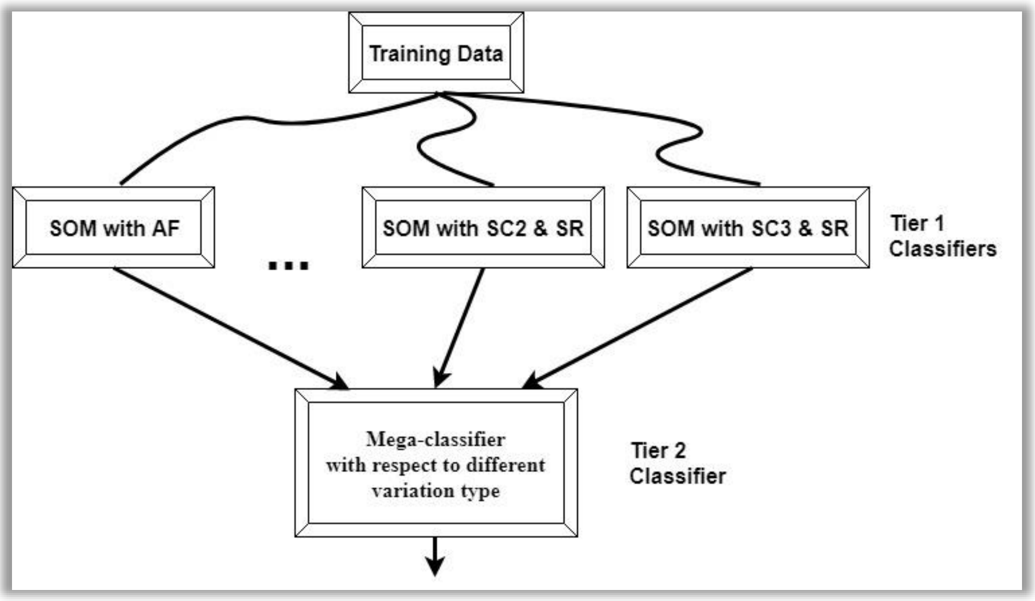


*Figure 5.1-2. RL-based EML Mechanism: 'Stacked Generalization'.*

The upper part of Figure 5.1-5 is the SOM-based Neural Network where we treat each original music melody (MDL and T-MML) as a hidden neuron, $T_n()$, and the input melody, $C()$, which either be the melody from the Training Data in Figure 5.1-2 & Figure 5.1-3 or as a general melody. The original music with highest similarity score will be the output. I.e., the one highlighted in yellow.

During each comparison  between the current input, $C()$, and the hidden neurons, $T_n()$, denoted as "*", different routes may have been chosen for evaluating the most 'suitable' similarity score, in other words, this is the different actions MUCASM takes against different variation type labels. The lower part of Figure 5.1-5 illustrates the process flow. The ones highlighted in red correspond to the 'Similarity-based similarity measurement' from Figure 5.1-1 and the ones highlighted in blue correspond to the 'Audio-based similarity measurement' from Figure 5.1-1. The strategy tree generated in Figure 5.1-3 is based on this process flow.

Finally, Algorithm 8 shows the EML using MDL and MML where $MDL\&T\text{-}MML_{Melody_{ori,var}}$ is defined as $(MDL_{Note} \quad T\text{-}MML_{MDL})_i$ over the index of the original melody and the index of the variation type, and i is the note index of the melody. Again, MDL is either defined as S-MDL, A-MDL, or the combination form in Section 4.5.

*Algorithm 8. RL-based EML for MDL&MML*

**Input:** MDL&MML self-training sets, $MDL\&MML_{Melody_{ori,var}}$.

**Output:** Accuracy for the self-testing, $acc$

```
1. Read $MDL\&MML_{Melody_{ori,var}}$ and parameter initializations (e.g., learning rate, $\eta$ and number of iterations $k$).
2. Define hidden neurons from $MDL\&MML_{Melody_{ori,var}}$, $h_p$, where $p$ is the original melody's index.
   $\mathcal{H} = \{h_p,\ \forall p \in \mathbb{N}\} \subseteq \mathcal{M} = \{MDL\&MML_{Melody_{ori,var}},\ \forall ori, var \in \mathbb{N}\}$
% Self-training.
3. For iteration = 1 to k
   a. For each input from $MDL\&MML_{Melody_{ori=x,var=y}}$.
      i. Normalization over all $h_p$ using modified Equation (30).
      ii. Evaluate Similarity score over all $h_p$.
          $Sim(MDL_{1,i}(j), MDL_{2,i}(j)) := \begin{cases} w_C * Sim_C + w_R * Sim_R + b \\ or \\ Sim_H \end{cases}$
          where $Sim_C$ can choose either SC1, SC2 or SC3, $Sim_R$ is RC followed by SR and $Sim_H$ is using AF & Algorithm 6.
      iii. $Sim_{max} = Sim(MDL\&MML_{Melody_{x,y}}, h_{p_{max}}) = max\{Sim(MDL\&MML_{Melody_{x,y}}, h_p),\ \forall p \in \mathbb{N}\}$
                                                            (Activation function)
      iv. Return index $p_{max}$.
      v. Update $h_{p_{max}}$ using SL1 or modified SL2
         where $T_n = h_{p_{max}}, n = p_{max}$ and $C = MDL\&MML_{Melody_{x,y}}$, and the learning rate $\eta$.
   b. End for
4. End for
```

```
% For Value Generation after each epoch (Self-testing).
5. For each variation type
    a. For each route from the process flow (Figure 5.1-5)
        i. Repeat step 3.a.i upto 3.a.iv for all $MDL\&MML_{Melody_{ori,var}}$ in the training sets to obtain a confusion matrix.
        ii. $\widetilde{acc} = \frac{|\{(ori,var),\ p_{max}=ori\}|}{|ori|*|var|}$, from the confusion matrix.
        iii. Output $\widetilde{acc}$
        iv. Compare and get the max accuracy and the corresponding route index.
        v. Strategy Tree generation.
    b. End for
6. End for
% (Self-)testing.
7. Repeat step 3.a.i upto 3.a.iv for all $MDL\&MML_{Melody_{ori,var}}$ in the training sets to obtain a confusion matrix, based on the strategy tree.
8. $acc = \frac{|\{(ori,var),\ p_{max}=ori\}|}{|ori|*|var|}$, from the confusion matrix.
9. Output $acc$
```

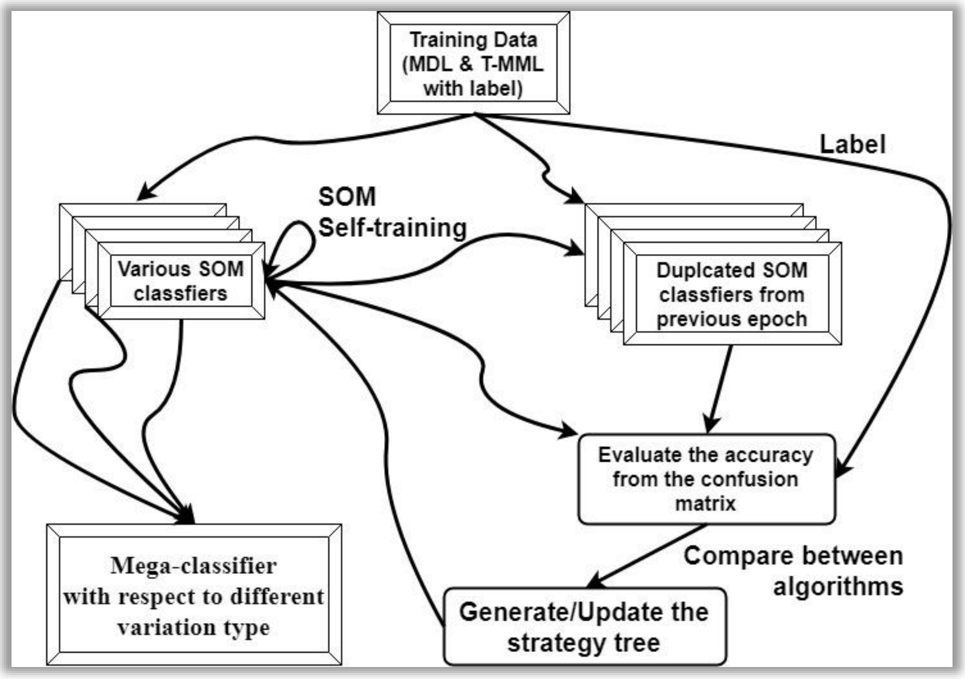

Training Data (MDL & T-MML with label)
Label
SOM Self-training
Various SOM classfiers
Duplcated SOM classfiers from previous epoch
Evaluate the accuracy from the confusion matrix
Compare between algorithms
Mega-classifier with respect to different variation type
Generate/Update the strategy tree

*Figure 5.1-3. RL-based EML Mechanism: 'Comparison between algorithms'.*

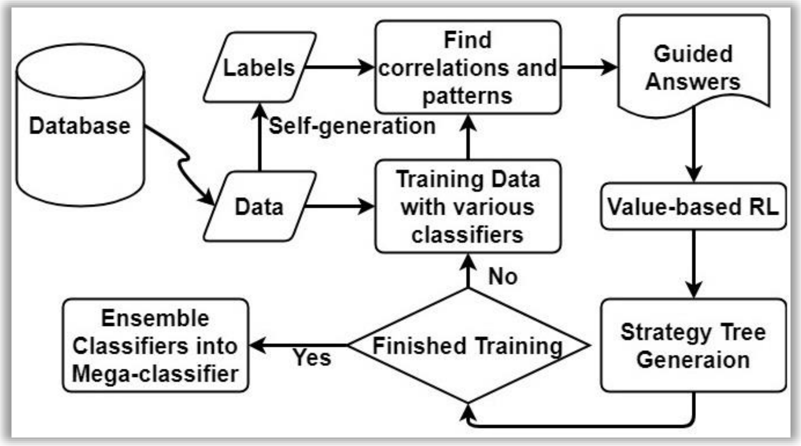

Database
Labels
Find correlations and patterns
Guided Answers
Self-generation
Data
Training Data with various classifiers
Value-based RL
No
Ensemble Classifiers into Mega-classifier
Yes
Finished Training
Strategy Tree Generaion

*Figure 5.1-4. The mechanism for the self-supervised learning.*

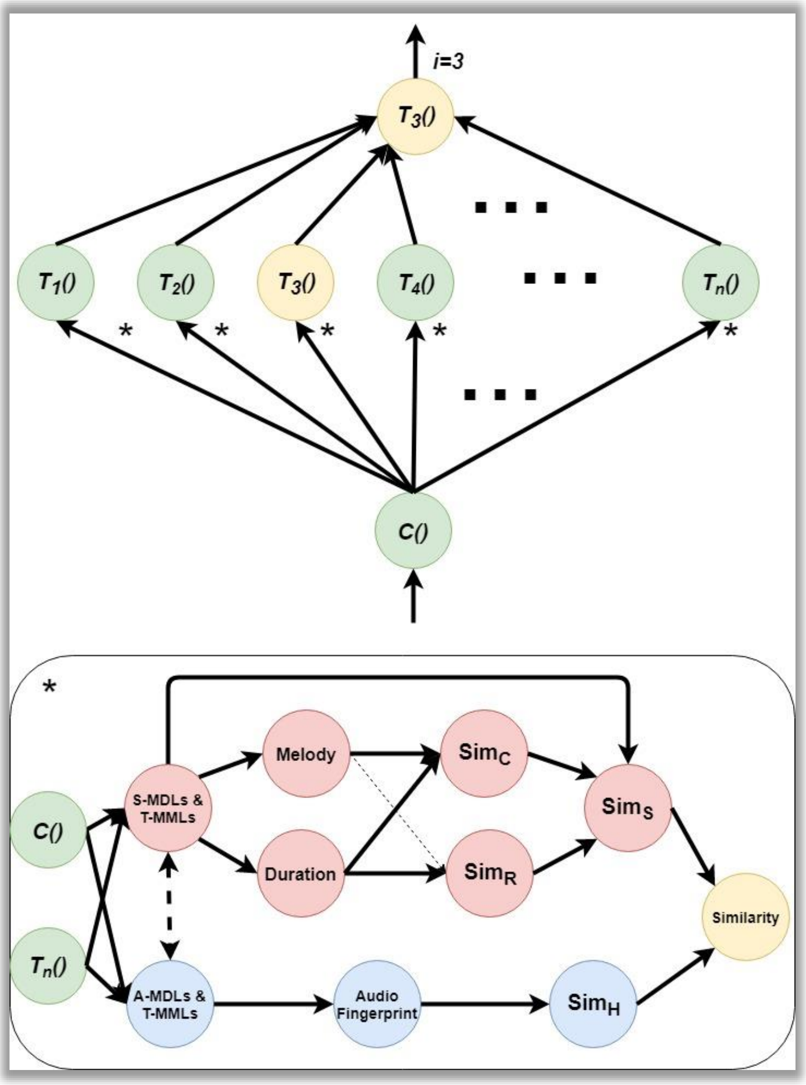


*Figure 5.1-5. RL-based EML Mechanism: 'SOM' and the 'process flow' for the Strategy Tree.*

## 5.2. Experiment, Result and Discussion

With the design of MUCASM, we carried out several experiments to evaluate the possibility and potentialbility of the proposed MUCASM mechanism. The experiments are all based on ten original monophornic melodies, music track [23] to [32]. We selected the melodies such that they are representative.

Each original melody considered four bars. We have manually generated the following four variation types: ‘Rhythm Variation’, where we have altered the rhythm; ‘Key Variation’, where we have altered the musical key; ‘Expansion’, where we have expanded melody by inserting some extra notes onto the original melody; ‘Reduction’ where we have reduced the melody which is literally by removing some notes. In future, these labels would be automatically generated by the computer to satisfy the self-supervised learning setting. However in this section, we manually input these labels using numbers without let the computer know the exact meaning to test the mechanism.

In addition, Track [24] has modified to $\frac{4}{4}$, rather than $\frac{5}{4}$. Hence, there are 50 S-MDL sequences in total to start with, i.e., the input of the MUCASM is the 50 vector-based sequences which have been treated as matrices. In other words, with the resolution of $\frac{1}{16}$, 3,200 vectors are considered. Alternatively, 250 streams (time series) are generated, as we removed $A_d$ from the definition of S-MDL, because they are all 1s. We believe they are representative. At later stage, there are 50 MDL (combination of A-MDL and S-MDL, refer to Section 4.5) and T-MML sequences where by the A-S Theorem, the data is theoretically equivalent. Thus, the file doubled to 50 two-column matrix sequence. We manually created both files in Excel format as we are focused on the impact of MDL&MML on MUCASM for OMI at this point, and it would be easier for MatLab to import.

The first experiment is based on the SOM, where we evaluate the best choice of similarity calculation and updating formula, so we tested with various combinations of those equations. The second experiment is based on the EML, where we evaluate how much advantage we can have by combining those methods, as well as through generating a strategy tree to describe the relationship between the variation type and the choice of those equations.

### 5.2.1. Experiment I: SOM

This section discusses the experiments based on the Kohonen Self-Organized Map Unsupervised Neural Network only for the OMI.

Our first experiment was carried out as follows: using SC1, SC2, SC3 with SL1 and SL2. Modify any factor that could affect the performance of the Self-Organized Map.

Let $k = 100$, $\eta = 0.0001$, $w_M = w_D = 0.5$ and $b = 0$, we have Table 5.2-1 for the accuracies from the confusion matrix of the 50 melodies.

| **Similarity Calculation** | **Self-Learning Method** | **Accuracy (Rating) (Before SOM)** | **Accuracy (Rating) (After SOM)** |
|---|---|---|---|
| SC1 | SL1 | 0.84 | 0.40 |
| SC1 | SL2 | 0.84 | 0.24 |
| SC2 | SL1 | 0.62 | 0.74 |
| SC2 | SL2 | 0.62 | 0.72 |
| SC3 | SL1 | 0.82 | 0.62 |
| SC3 | SL2 | 0.82 | 0.76 |

*Table 5.2-1. Priliminary SOM result table.*

From these preliminary results, we filtered SC1 as the accuracy drops massively (over 50%) after SOM and does not have much advantages without SOM. We also filtered the combination: SC3 and SL1 for the same reason. However, we keep the combination: SC3 and SL2 because it has obtained the highest accuracy after SOM, despite the accuracy dropping after SOM. Hence, we rename the remaining three combinations as following: S21, S22 and S32 repectively.

Next, we tested the effectiveness of using SOM.

Let $k = 1000$ and keep $k$ as a constant, we have altered $\eta$ to be 0.0001, 0.001, 0.01, and 0.05. Similarily, let $\eta = 0.001$ and keep $\eta$ as a constant, we have altered $k$ to be 100, 1000, 5000 and 10000. In theory, $w_M$ and $w_D$ will be updated after each epoch, another loop outside the existing iteration loop in Algorithm 4. However, we have captured the accuracies for the following ratios for the explanation purpose: $0.1{:}\,0.9$; $0.3{:}\,0.7$; $0.5{:}\,0.5$; $0.7{:}\,0.3$; and $0.9{:}\,0.1$ ($w_M{:}\,w_D$). The results have shown in Table 5.2-2 and Table 5.2-3.

| Equation | Initial | Accuracy After SOM ($k = 1000$) | | | |
|---|---|---|---|---|---|
| ($w_M$: $w_D$) | Accuracy | $\eta = 0.0001$ | $\eta = 0.001$ | $\eta = 0.01$ | $\eta = 0.05$ |
| **S21** | 0.62 | 0.74 | 0.74 | 0.74 | 0.74 |
| **S22** | 0.62 | 0.72 | 0.74 | 0.74 | 0.74 |
| **S32($0.1$: $0.9$)** | 0.86 | 0.14 | 0.02 | 0.12 | 0.20 |
| **S32($0.3$: $0.7$)** | 0.84 | 0.72 | 0.40 | 0.34 | 0.68 |
| **S32($0.5$: $0.5$)** | 0.82 | 0.64 | 0.64 | 0.66 | 0.68 |
| **S32($0.7$: $0.3$)** | 0.78 | 0.68 | 0.62 | 0.66 | 0.66 |
| **S32($0.9$: $0.1$)** | 0.78 | 0.74 | 0.70 | 0.64 | 0.64 |

*Table 5.2-2. SOM with $k$ as a constant (contour melody only).*

| Equation | Initial | Accuracy After SOM ($\eta = 0.001$) | | | |
|---|---|---|---|---|---|
| ($w_M$: $w_D$) | Accuracy | $k = 100$ | $k = 1000$ | $k = 5000$ | $k = 10000$ |
| **S21** | 0.62 | 0.74 | 0.74 | 0.74 | 0.74 |
| **S22** | 0.62 | 0.72 | 0.74 | 0.74 | 0.74 |
| **S32($0.1$: $0.9$)** | 0.86 | 0.24 | 0.02 | 0.02 | 0.08 |
| **S32($0.3$: $0.7$)** | 0.84 | 0.72 | 0.40 | 0.40 | 0.52 |
| **S32($0.5$: $0.5$)** | 0.82 | 0.64 | 0.64 | 0.66 | 0.64 |
| **S32($0.7$: $0.3$)** | 0.78 | 0.68 | 0.64 | 0.66 | 0.66 |
| **S32($0.9$: $0.1$)** | 0.78 | 0.74 | 0.70 | 0.64 | 0.64 |

*Table 5.2-3. SOM with $\eta$ as a constant (contour melody only).*

From these two tables, we have made the following observations:

- With SC2, Self-Organized Map helps to improve the accuracy by $\frac{(0.74-0.62)}{0.62} = 19.4\%$.
- For SC3, high accuracy obtained with pre-defined original music melodies as hidden neurons. However, the accuracy drops after the training. This is due to the fact that we have altered the melody itself. On the other hand, theoretically, there is no guarantee that the exact original theme has been chosen for the hidden layer neurons.
- Despite the drop in accuracy with SC3, we can still observe that, for different weight ratio ($w_M$: $w_D$), the melody preference similarity measurement, $w_M > w_D$, can give more stable accuracy. Overall, $w_M$: $w_D = 0.7$: $0.3$ is the most stable one, with an initial accuracy rating of 0.78 and 0.66 after applied SOM.

Stable means changing in the learning rate or the number of iterations has less effect on the accuracy after the self-training. This matches our musical knowledge, contouric melody flow affects more than the duration of the music.

With these findings, our next experiment has been set up such that we can see the performance of SOM by introducing another feature: the rhythm (RC, SR).

| **Equation** | **Initial** | **Accuracy After SOM ($k = 1000$)** | | | |
|---|---|---|---|---|---|
| **($w_C : w_R$)** | **Accuracy** | $\eta = 0.0001$ | $\eta = 0.001$ | $\eta = 0.01$ | $\eta = 0.05$ |
| **S21$(0.5: 0.5)$** | 0.90 | 0.84 | 0.84 | 0.84 | 0.84 |
| **S21$(0.7: 0.3)$** | 0.92 | 0.84 | 0.84 | 0.82 | 0.84 |
| **S21$(0.9: 0.1)$** | 0.94 | 0.82 | 0.82 | 0.78 | 0.78 |
| **S22$(0.5: 0.5)$** | 0.90 | 0.72 | 0.62 | 0.70 | 0.62 |
| **S22$(0.7: 0.3)$** | 0.92 | 0.82 | 0.76 | 0.76 | 0.78 |
| **S22$(0.9: 0.1)$** | 0.94 | 0.72 | 0.62 | 0.70 | 0.62 |
| **S32$(0.5: 0.5)$** | 0.80 | 0.70 | 0.60 | 0.58 | 0.56 |
| **S32$(0.7: 0.3)$** | 0.80 | 0.68 | 0.58 | 0.58 | 0.58 |
| **S32$(0.9: 0.1)$** | 0.78 | 0.66 | 0.56 | 0.66 | 0.66 |

*Table 5.2-4. SOM with $k$ as a constant (contour melody and rhythm, $w_M: w_D = 0.7: 0.3$).*

| **Equation** | **Initial** | **Accuracy After SOM ($\eta = 0.001$)** | | | |
|---|---|---|---|---|---|
| **($w_C : w_R$)** | **Accuracy** | $k = 100$ | $k = 1000$ | $k = 5000$ | $k = 10000$ |
| **S21$(0.5: 0.5)$** | 0.90 | 0.84 | 0.84 | 0.84 | 0.84 |
| **S21$(0.7: 0.3)$** | 0.92 | 0.84 | 0.84 | 0.84 | 0.84 |
| **S21$(0.9: 0.1)$** | 0.94 | 0.82 | 0.82 | 0.82 | 0.82 |
| **S22$(0.5: 0.5)$** | 0.90 | 0.72 | 0.62 | 0.60 | 0.60 |
| **S22$(0.7: 0.3)$** | 0.92 | 0.82 | 0.76 | 0.76 | 0.76 |
| **S22$(0.9: 0.1)$** | 0.94 | 0.82 | 0.80 | 0.80 | 0.80 |
| **S32$(0.5: 0.5)$** | 0.80 | 0.58 | 0.60 | 0.60 | 0.60 |
| **S32$(0.7: 0.3)$** | 0.80 | 0.68 | 0.58 | 0.58 | 0.58 |
| **S32$(0.9: 0.1)$** | 0.78 | 0.66 | 0.56 | 0.60 | 0.62 |

*Table 5.2-5. SOM with $\eta$ as a constant (contour melody and rhythm, $w_M: w_D = 0.7: 0.3$).*

| Accuracy Summary / Features / Equation<br>$(w_M : w_D)(w_C : w_R)$ | Contour ☑<br>Rhythm ☐ | Contour ☑<br>Rhythm ☑ |
|---|---|---|
| **S21**<br>**(n/a)**<br>$\mathbf{(0.5:0.5)}$ | 0.74<br>$(max = 0.74)$<br>$(Initial = 0.62)$ | 0.84<br>$(max = 0.84)$<br>$(Initial = 0.90)$ |
| **S21**<br>**(n/a)**<br>$\mathbf{(0.7:0.3)}$ | | 0.8375<br>$(max = 0.84)$<br>$(Initial = 0.92)$ |
| **S21**<br>**(n/a)**<br>$\mathbf{(0.9:0.1)}$ | | 0.81<br>$(max = 0.82)$<br>$(Initial = 0.94)$ |
| **S22**<br>**(n/a)**<br>$\mathbf{(0.5:0.5)}$ | 0.735<br>$(max = 0.74)$<br>$(Initial = 0.62)$ | 0.65<br>$(max = 0.72)$<br>$(Initial = 0.90)$ |
| **S22**<br>**(n/a)**<br>$\mathbf{(0.7:0.3)}$ | | 0.7775<br>$(max = 0.82)$<br>$(Initial = 0.92)$ |
| **S22**<br>**(n/a)**<br>$\mathbf{(0.9:0.1)}$ | | 0.8125<br>$(max = 0.86)$<br>$(Initial = 0.94)$ |
| **S32**<br>$\mathbf{(0.7:0.3)}$<br>$\mathbf{(0.5:0.5)}$ | 0.66<br>$(max = 0.68)$<br>$(Initial = 0.78)$ | 0.6025<br>$(max = 0.70)$<br>$(Initial = 0.80)$ |
| **S32**<br>$\mathbf{(0.7:0.3)}$<br>$\mathbf{(0.7:0.3)}$ | | 0.605<br>$(max = 0.68)$<br>$(Initial = 0.80)$ |
| **S32**<br>$\mathbf{(0.7:0.3)}$<br>$\mathbf{(0.9:0.1)}$ | | 0.6225<br>$(max = 0.66)$<br>$(Initial = 0.78)$ |

*Table 5.2-6. Comparison and summarization result table for the average accuracy with and without the Rhythm feature.*

For this experiment, we concentrated on the ratio $w_C : w_R$ while maintaining $w_M : w_D = 0.7 : 0.3$ and filtered out the results with $w_C < w_R$. Thus, in Table 5.2-4, $k$ has been kept as a constant value of 1000 and altered $\eta$ to be 0.0001, 0.001, 0.01, and 0.05, whereas in Table 5.2-5, $\eta$ has been kept as a constant value of 0.001 and altered $k$ to be 100, 1000, 5000 and 10000. Based on Table 5.2-2 ~ Table 5.2-5, Table 5.2-6 summarizes the accuracy with or without the rhythm feature.

From these tables, we have made the following observations:

- By adding a new musical feature, the accuracy for the MUCASM increases. Especially, with SC2 and pre-defined original music melodies, the maximum potential accuracy is 0.94.
- The optimal weight ratio $w_C : w_R$ is around 0.7:0.3 and 0.9:0.1. In other words, the melody flow is much more important compared to the rhythm. This is the reason that we can mashup two original music tracks which have similar tempo in Section 1.1, and does not escalate them up to a plagiarism cases.

Additionally, Table 5.2-7 has shown some further analysis on the type of variations which will mostly like benefit from SOM, based on two typical results from Table 5.2-2 ~ Table 5.2-5.

| Equation / SOM Status / Variation Type | S21(n/a)$(0.7:0.3)$ | | S32$(0.7:0.3)(0.7:0.3)$ | |
|---|---|---|---|---|
| | Intial Accuracy | Accuracy After SOM $(\eta = 0.001)$ $(k = 1000)$ | Initial Accuracy | Accuracy After SOM $(\eta = 0.001)$ $(k = 1000)$ |
| **Original** | 1.00 | 1.00 | 1.00 | 0.70 |
| **Rhythm Variation** | 0.70 | 0.90 | 0.90 | 0.80 |
| **Key Variation** | 1.00 | 0.40 | 0.30 | 0.10 |
| **Expansion** | 1.00 | 1.00 | 1.00 | 0.70 |
| **Reduction** | 0.90 | 0.90 | 0.80 | 0.60 |
| **Overall** | 0.92 | 0.84 | 0.80 | 0.58 |

*Table 5.2-7. Comparison and summarization result table for detailed variational type breakdowns.*

Table 5.2-7 has shown that ‘Rhythm Variation’ is more likely to be benefit from the SOM mechanism, as the accuracy increased to 0.90 from 0.70. This suggests that if we consider different route for each variation type, i.e., some variation type using the pre-training data and some using the post-training data, the overall performance will be increased as we combined the maximum accuracy of individual type with different route.

Moreover, the ‘Key Variation’ can achieve 1 for SL21, which is an improvement from the existing audio-based MIR system. Meanwhile, the ‘Expansion’ and ‘Reduction’ variation types obtained a potential accuracy of 1 and 0.9 respectively, which improves from the existing symbolic-based MIR system. On the other hand, a high accuracy rate of 0.90 has obtained for the ‘Rhythm Variation’ by altering the similarity measurement and the learning method. This suggested various similarity measurements and learning methods can be the route options too.

### 5.2.2. Experiment II: RL-based EML

This section shows the results from the experiments which is based on the RL-based Ensembled Machine Learning for the OMI.

Our first test only considered A-MDL with the modified Equation (25) ~ (31) described in Section 5.1.2.2. Table 5.2-8 shows two typical results.

| | **Initial Accuracy** | **Accuracy After SOM $(\eta = 0.001)(k = 1000)$** |
|---|---|---|
| **S21(n/a)$(0.7: 0.3)$** | 0.80 | 0.40 |
| **S22(n/a)$(0.7: 0.3)$** | 0.80 | 0.14 |
| **S31$(0.7: 0.3)(0.7: 0.3)$** | 0.86 | 0.52 |
| **S32$(0.7: 0.3)(0.7: 0.3)$** | 0.86 | 0.20 |

*Table 5.2-8. SOM with $\eta, k$ as a constant, considered A-MDL only and modified equations.*

From Table 5.2-8 and comparing with Table 5.2-6, we can made the following observarions:

- For A-MDL, SL1 is better than SL2 when SOM is applied, whereas SC3 is better than SC2 for similarity measurement.

- SCs and SLs gave better overall good measurements and learning mechanisms for S-MDL rather than A-MDL. For example, the two accuracies for S21(n/a)(0.7: 0.3) with S-MDL were 0.92 and 0.84, but 0.80 and 0.40 for A-MDL.

These comparisons suggested that SCs and SLs were both designed for symbolic-based similarity measurement.

Our next test considered the combination form of MDL and its corresponding T-MML, as described in Section 4.5, and the audio fingerprint approach, $AF$. The results have shown in Table 5.2-9 where 'A' corresponds to $\begin{cases} \boldsymbol{H(x,2) \leftarrow t(f_i) = 1} \\ \boldsymbol{H(x,2) \leftarrow t(f_i) = t(f_i) + 1} \end{cases}$ and 'B' corresponds to $\begin{cases} \boldsymbol{H(x,2) \leftarrow t(f_i) = D_i} \\ \boldsymbol{H(x,2) \leftarrow t(f_i) = t(f_i) + D_i} \end{cases}$ from Algorithm 5.

From these results, we have made the following observations:

- On average, use the compression ratio of 50% is better than using 100% as in Table 5.2-9 most accuracy rating with $C = 50\%$ is larger than $C = 100\%$.
- If the sampling rate is less than 1, the hash table calculation method 'B' is better than 'A' and if is greater than 1, currently, 'A' is better than 'B'. However, there is a conjectured trend of increasing in accuracy for 'B' whereas 'A' almost obtained a stable accuracy of 0.66.

Therefore, we pick the sampling rate of 1000%, i.e., enlarge the original MDL length to 10 times longer, the compression rate of 50% and 'B' for our final test.

Our final experiment tests the RL-based EML where we combined both audio and symbolic approaches, and the value is based on the accuracy from the confusion matrix. We have adapted different similarity measurements for different variation types and generated a strategy tree, to increase the overall accuracy. After 1000 iterations with $\eta = 0.001$ (defined to be one epoch), we run a self-testing to evaluate the strategy tree, to see which route for different type of variations maximizes the final accuracy. Thus, theoretically, this change of routing information will affect the choice of maximum hidden neurons for the next epoch.

Typical results have shown in Table 5.2-10 with the comparison of using single (audio or symbolic) approach and using both (audio and symbolic) approaches.

| AF-Equation (Sampling Rate) (Compression Rate) | SOM Status A/B | Initial Accuracy | Accuracy After SOM $(\eta = 0.001)$ $(k = 1000)$ |
|---|---|---|---|
| AF-SL1/2 $(R = 50\%)$ $(C = 50\%)$ | A | 0.64 | 0.56 |
| | B | 0.70 | 0.54 |
| AF-SL1/2 $(R = 50\%)$ $(C = 100\%)$ | A | 0.64 | 0.48 |
| | B | 0.68 | 0.52 |
| AF-SL1/2 $(R = 200\%)$ $(C = 50\%)$ | A | 0.72 | 0.66 |
| | B | 0.72 | 0.62 |
| AF-SL1/2 $(R = 200\%)$ $(C = 100\%)$ | A | 0.66 | 0.66 |
| | B | 0.68 | 0.56 |
| AF-SL1/2 $(R = 500\%)$ $(C = 50\%)$ | A | 0.72 | 0.66 |
| | B | 0.72 | 0.62 |
| AF-SL1/2 $(R = 500\%)$ $(C = 100\%)$ | A | 0.66 | 0.66 |
| | B | 0.68 | 0.56 |
| AF-SL1/2 $(R = 1000\%)$ $(C = 50\%)$ | A | 0.72 | 0.66 |
| | B | 0.74 | 0.64 |
| AF-SL1/2 $(R = 1000\%)$ $(C = 100\%)$ | A | 0.66 | 0.66 |
| | B | 0.68 | 0.58 |

*Table 5.2-9. SOM with audio fingerprint approach only, considered MDL & T-MML.*

| Equation Labels $(R)$&$(C)$&A/B $(w_M: w_D)(w_C: w_R)$ / Method / SOM Status | | Initial Accuracy | Accuracy After SOM $(\eta = 0.001)$ $(k = 1000)$ |
|---|---|---|---|
| **(AF+SC2)*SL1 1000%&50%&B (n/a)$(0.7: 0.3)$** | Audio | 0.74 | 0.64 |
| | Symbolic | 0.92 | 0.84 |
| | EML | 0.96 | 0.84 |
| **(AF+SC2)*SL2 1000%&50%&B (n/a)$(0.7: 0.3)$** | Audio | 0.74 | 0.64 |
| | Symbolic | 0.92 | 0.76 |
| | EML | 0.96 | 0.76 |
| **(AF+SC3)*SL1 1000%&50%&B $(0.7: 0.3)(0.7: 0.3)$** | Audio | 0.74 | 0.64 |
| | Symbolic | 0.80 | 0.64 |
| | EML | 0.80 | 0.68 |
| **(AF+SC3)*SL2 1000%&50%&B $(0.7: 0.3)(0.7: 0.3)$** | Audio | 0.74 | 0.64 |
| | Symbolic | 0.80 | 0.58 |
| | EML | 0.80 | 0.66 |
| **AF*SL2+(SC2+SC3)*SL1 1000%&50%&B $(0.7: 0.3)(0.7: 0.3)$** | Audio | 0.74 | 0.64 |
| | Symbolic | 0.92 | 0.84 |
| | | 0.80 | 0.64 |
| | EML | 0.96 | 0.84 |

*Table 5.2-10. EML with MDL & T-MML.*

From these results, we have made the following observations:

- By using the 'Stacked Generalization' and 'Comparison between algorithms', the accuracy can be improved.
- It will be better for 'Original' and 'Rhythm Variation' to use the Audio Fingerprint approach, whereas the Tier 1 SOM classifier using SL2 at training. In the meanwhile, 'Key Variation' and 'Reduction' to use the symbolic approach with SC2 and 'Expansion' is with SC3, whereas the Tier 1 SOM classifiers using SL1 at training.

Figure 5.2-1 shows the hidden strategy tree for the final mega-classifier, learned from our RL-based EML MUCASM mechanism.

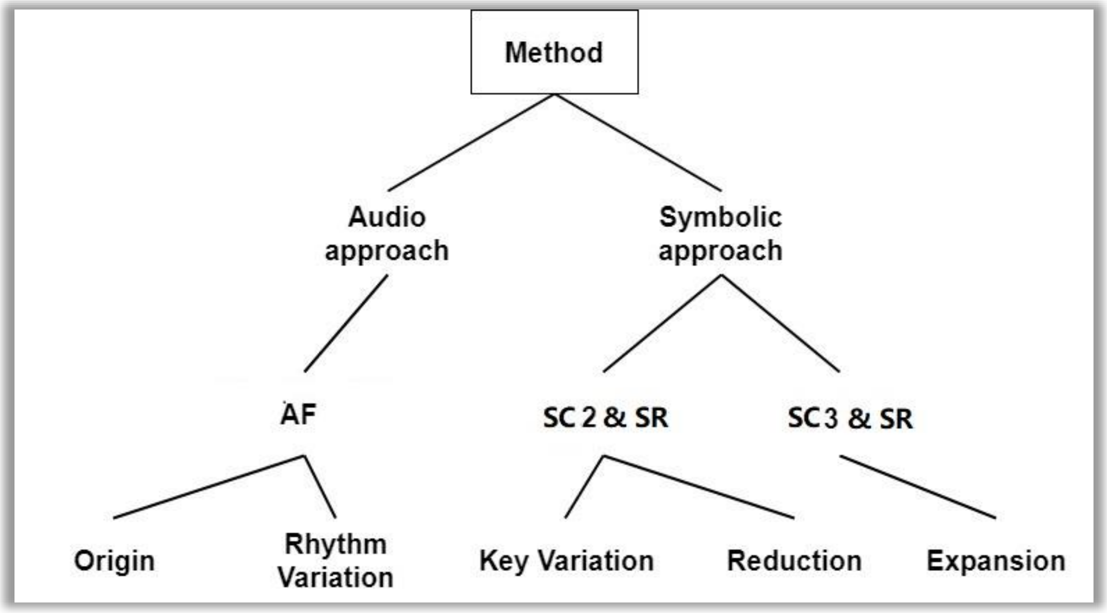


*Figure 5.2-1. Strategy tree generated by the RL-based EML Mechanism.*

## 5.3. Summary

From all the experiments in this chapter, we have illustrated that the proposed MUCASM mechanism allows Original Music Identifier more accurate, as by adding features, ensembling different algorithms (with various similarity measurements and update formulae) and introducing self-supervised learning mechanism, there is an increase in the accuracy rating of the classification problem, from 74% to 96%.

Thus, the next chapter evalutes the performance against existing MIR systems.

## 6. Further Discussion and Evaluation

With those proof-of-concept prototypes for the proposed coding scheme and the MUCASM mechanism, we now need to further evaluate the E$^3$MSD architecture, based on the summaries from the literature review and the research objectives.

### 6.1. MDL&MML Conversions

We have modified Algorithm 2 and Algorithm 3 such that the input is a MIDI file instead. Hence, we can evaluate our signal processing prototype implemented in Section 4.5 as well as demonstrate the relationship between MDL and MIDI files described in Section 4.1, by comparing the parameter values of the melody in MIDI format and in the converted (S-)MDL format. The procedure outline is shown in Algorithm 9.

***Algorithm 9. MIDI MDL Conversion***

```
Input: MIDI File
Output: A-MDL/(S-)MDL Files, Similarity scores: $Sim_C$ & $Sim_R$.
    1. Read MIDI files and parameter initializations (e.g., $\theta$ and $\vartheta$)
    2. Fetch the acoustic value.
    % line 3-6 were based on Algorithm 2.
    3. Extract relevant features using FFT.
    4. Convert into A-MDL file.
    5. Convert into S-MDL file, rounding when necessary.
    6. Output A-MDL and S-MDL files.
    % line 7-8 were based on Algorithm 3.
    7. Extract the melody line, $m_1(t)$ & $m_2(t)$, and the rhythm line, $r_1(t)$ & $r_2(t)$,
       from the relevant parameter values of the MIDI file and the converted S-
       MDL file, respectively.
    8. Evaluate and output $Sim_C$ & $Sim_R$ using line 693-12 of Algorithm 3.
```

After input track [27], in MIDI format, to Algorithm 9 and set $\theta = -1, \vartheta = 1.0\,Hz$, $\mathrm{Sim}_C = 85.93\%$ for A-MDL and $\mathrm{Sim}_C = 100\%$ for S-MDL, whereas $\mathrm{Sim}_R = 100\%$ for both MDL files. This result has further indicated that S-MDL is corresponding to MIDI, and hence the assumption made in Section 5.1 is reasonable.

On the other hand, as already mentioned in Section 4.3, the further evaluation for the effectiveness of MDL and T-MML to the audio sound is included in Xue's project (Xue, 2018).

## 6.2. MUCASM

To evaluate MUCASM, we input twenty melodies onto the trained MUCASM after first epoch. Half of them were those origin pieces and the remaining half were 'new' melodies, such that the first half were used to test the exact search and the second half were used to test the approximate search. Those ten 'new' melodies do not belong to our training dataset and randomly covers each variation type from those ten origins. They were manually implemented and converted into the MDL and MML format.

For the exact search, from original to original, it has obtained 100% accuracy for all Tier-1 classifiers and the final mega-classifier, which can be achieved by the audio-based MIR systems using the same melody segment. However, the existing audio-based MIR systems were limited in approximate search, from variation to original, which has described in Section 2.1.1, but our MUCASM is capable, as we have found out that approximately 80% of those 'new' melodies have successfully found its origin by our MUCASM.

Furthermore, we also compared with Musipedia, a symbolic-based MIR system. Due to the limitation of the music that is stored in Musipedia, more importantly, some of the melodies in our database are 'fake' variation music melodies. It is hard to get a direct comparison for benchmarking. Therefore, we evaluate the melodies that exist in both database.

After testing the music data on Musipedia by translating our MDL to their input, we found out that the music piece "Canon" and "Twelve Variations On "Ah Vous Dirai-Je, Maman" were stored in the Musipedia Database. For the "Canon" melody, Musipedia failed to retrieve the origin from the 'Expansion' and 'Reduction' variations. Meanwhile, for the

“Twelve Variations On "Ah Vous Dirai-Je, Maman” melody, instead of “Twinkle, Twinkle, Little Star”, we cannot retrieve the complex version from the simplified melody, which further confirmed that Musipedia cannot deal with the ‘Reduction’ variation. By comparing the result with those ten ‘new’ melodies, MUCASM has obtained a relative higher retrieval accuracy.

Finally, for the processing time for our MUCASM, it is going to be $O(e * k * n * m * t)$ for training as in Algorithm 8, we loop melodies which either are the origin or the variations, $n$, and the number of origin melodies, $m$ with respect to the number of variation types , $t$, for $e * k$ times, where $e$ is the number of epoches and $k$ is the number of iterations. For well-trained MUCASM, the search order is just $O(m)$. However, we have not considered the search using the melody segement from a long melody that is stored in the database.

## 6.3. Evaluation

Table 6.3-1 shows how much we have covered in our E$^3$MSD prototypes against the features and limitations summarized in Section 2.1.1 and 2.1.2, and the designed features in Chapter 3, which also correspond to the research objectives described in Section 1.3.

The followings are the notes for Table 6.3-1 where there is no space underneath the table:

[1] In Excel format.

[2] (Xue, 2018).

[3] Almost no variations and noise sensible.

[4] Not good at ‘Expansion’ and ‘Reduction’.

[5] Potentially High.

| Research Objectives/Features/Limitations | | | Audio-based | Symbolic-based | $E^3$MSD-based |
|---|---|---|---|---|---|
| Code ↓ **Music** | Literature Review | | ✓ | ✓ | n/a |
| | Fedility of Music | Normal note | ✓ | ✓ | ✓ |
| | | Glissando | Continuous | Discrete | Continuous |
| | | Lyrics | ✓ | ✗ | ✗ |
| | File Size | | Large | Small | Medium[1] |
| | Proof-of-concept | | n/a | n/a | ✓, SPD Player[2] |
| Music ↓ **Code** | Extract/Represent musical features | Melody | Indirect | Direct | Direct |
| | | Rhythm | Indirect | Direct | Direct |
| | Conversion from Audio | | n/a | ✓ | ✓ |
| | Proof-of-concept | | n/a | n/a | ✓, FFT |
| Code ↓ **Similarity Scores** | Similarity Calculation methods | | Audio-Fingerprint (I.e., AF) | Various methods | Combined (AF, SL1, SL2, SL3, SR) |
| Applications | Proof-of-concept | | n/a | n/a | ✓, RL-EML |
| | Benchmark | | n/a | n/a | Indirect |
| | Size of dataset, for training and testing | | Huge | Large | Small |
| | Retrieving Types | | Genre, Melody/Lyrics, Mood, Timber | Melody, Rhythm, Scale, Contour | Melody, Rhythm |
| | Process Time | | Long | Short | Medium |
| | Retrieving Accuacy | Exact | High | High | High |
| | | Approximate | Low[3] | Medium[4] | High[5] |

*Table 6.3-1. Project Evaluation Summary Table.*

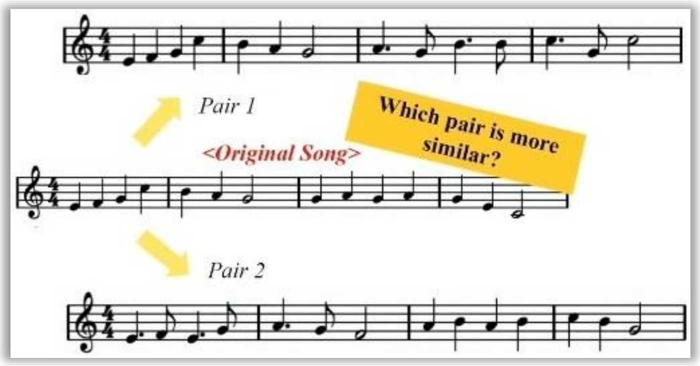


*Figure 6.3-1. Park et al. (2019)'s example.*

In addition, we use an example from Park et al. (2019), Figure 6.3-1, to illustrate the advantages of using MDL&MML. By using Park's approach, the melody pair 1 gave a similarity score of 0.7755 and the melody pair 2 gave 0.6815. i.e., the melody pair 1 has a higher similarity score than the melody pair 2. However, when she presented her paper in the ISMIR conference*, majority of the audience thought the opposite. If using MDL&MML with different types of MDL (S-MDL or A-MDL) and similarity measurements, both results can be obtained under different scenarios. For example, by comparing $Sim_M$, we may obtain $e^{-751}$ against $e^{-512}$ using SrC3 or $e^{-175}$ against $e^{-320}$ if we ignore $MDL_{1,i}(2)$ from SC3 resepectively. Moreover, by comparing $Sim_R$, both melody pairs give $\frac{26}{32}$ using SR, where the resolution is $\frac{1}{8}$. Equal similarity scores suggest that the music search purely based on the rhythm obtained low retrieval accuracy (Section 2.1.2 on page 22).

Finally, here are the limitations of our project on E$^3$MSD:

- From Table 6.3-1, the most critical limitation for our MUCASM is the amount of variation types and the size of music dataset, for training and testing.
- Despite we have use Figure 6.3-1 as an example, a large-scale benchmarking with well-recognized music tracks is still needed to provide a much stronger evidence that using MDL&MML is better than using other symbolic formats, as well as using MUCASM on OMI is better than using other mechanisms.

* https://collegerama.tudelft.nl/Mediasite/Showcase/ismir2019/Presentation/4c035b43fe5b4272a89dc222da1485271d

# 7. Conclusions and Future Works

## 7.1. Conclusion

We have proposed a musical data model named Music Definition Language and Music Manipulation Language. MDL has a similar concept with existing symbolic coding schemes whereas MML is introduced such that we can obtain a closer sound waveform without affecting the musial melody flow.

For completion, we have shown the relationship between the audio format, symbolic format and MDL&MML, theoretically and practically. The file size of an MDL and MML file is smaller than the audio file whereas the fidelity of the music using the MDL and MML format is better than using those symbolic formats.

It has been demonstrated that by using the proposed data model (MDL&MML), we can employ a Self-supervised Reinforcement-based Ensemble Learning to improve the performance of MUCASM for OMI. By introducing a weighted similarity measurements which combines the contouric melody and rhythm, and by altering the updating formula, we have achieved similarity accuracy of 84%. By introducing the RL-based EML that schedules different routes for different variation types of the origins, a potential similarity accuracy of 96% can be reached.

In addition, we have compared with Park et al. (2019)'s example and shown that our approach can provide various results under different scenarios.

All the results indicate that E$^3$MSD can maintain high symbolized music fidelity, lead to extensibility in searchability and improved search performance by integrating the audio-based and the symbolic-based approaches on the music data model and the learning mechanism. Hence, we believe that our research outcomes will benefit the off-line or on-line Music Search and the Plagiarism Protection, as well as music data storage and data representation.

## 7.2. Future Works

Here are some suggested future works:

- For the representation of the MDL&MML files, we can add a stochastic function such that the sound wave generated will be closer to the audio sounds, as well as considering the harmony of the music note.
- For MUCASM, we can add more musical features onto the process flow in Figure 5.1-5, resulting in more Tier 1 classifiers for the mega-classifier. We can increase the music length and depth to make the MUCASM work for complex music pieces, e.g., EM, and further integrate with Policy-based Reinforcement Learning. Moreover, we can upgrade MUCASM by incoperating Generative Adversarial Network (GAN), introduced by Ian Goodfellow (Goodfellow, et al., 2014), or Recurrent Generative Adversarial Network (RGAN), introduced by Briot et. al. (2017). GAN and RGAN have been applied to general image processing and signal processing to improve the robustness of the classifications (Reed, et al., 2016; Briot, et al., 2017; Pascual, et al., 2017). Their idea is to generate several 'fake' images from the real ones whereas we generated 'fake' melodies, as previously mentioned.
- For applications, we need to design more syntax and rules of MDL and MML.
- For project evaluation, we need to increase the amount of music for training and testing, as well as carry out some large-scale benchmarking from well-recognized music tracks.
- For more industrial uses, we can extend to remix music generation based on various similarity scores as well as implement an MDL&MML player, with or without videos.

## Music Copyright and References

### Music Copyright

[1] Aurora (2015) *Runaway*. [online] available from <https://www.youtube.com/watch?v=d_HlPboLRL8> (©AURORA) [5 March 2019]

[2] Gemie and SawanoHiroyuki[nZk] (2015) *X.U.*. [online] available from <https://www.youtube.com/watch?v=s7Kg-bpFX5k> (©SawanoHiroyuki[nZk] Official YouTube Channel) [5 March 2019]

[3] Gruttmann, J. (2004) *Schnappi, Das Kleine Krokodil*. [online] available from <https://www.youtube.com/watch?v=ony4W9pDzbI> (©fritz51242) [5 March 2019]

[4] Lim Kim (2013) *All Right*. [online] available from <https://www.youtube.com/watch?v=ZJMWhJdjlpg> (©MYSTIC Entertainment) [5 March 2019]

[5] Lynnsha (2008) *Si Seulement*. [online] available from <https://www.youtube.com/watch?v=Iku5WvjFwBA> (©Elaine C.) [5 March 2019]

[6] Manson, M. (2001) *The Nobodies*. [online] available from <https://www.youtube.com/watch?v=qi5nTb-NRFU> (©Marilyn Manson) [5 March 2019]

[7] Mizuki, N. and SawanoHiroyuki[nZk] (2014) *Aliez*. [online] available from <https://www.youtube.com/watch?v=UGraUyuDcwU> (©Hiroyuki Sawano Works by Nigel) [5 March 2019]

[8] Shin (Outsider), O. and Deffinite Of Sunday 2pm (2009) *Value Of The Man*. [online] available from <https://www.youtube.com/watch?v=KU6iD6e9GAY> (©Mog Tempest) [5 March 2019]

[9] Shivaree (2002) *Snake Eyes*. [online] available from <https://www.youtube.com/watch?v=QbGUOoEDbQI> (©George) [5 March 2019]

[10] Sung, S. (2003) *Life's A Struggle*. [online] available from <https://www.youtube.com/watch?v=M_RXZ5DvbSE> (©quan) [5 March 2019]

[11] Wong, L. (2003) *Can You Feel My World*. [online] available from <https://www.youtube.com/watch?v=3JSmWvRUCUY> (©王力宏 Wang Leehom) [5 March 2019]

[12] Yui (2010) *Again*. [online] available from <https://www.youtube.com/watch?v=2DYYVp4QXew> (©Anime Is Life)[5 March 2019]

[13] Bach, J. S. (1723) *Two-Part Inventions No.4 (D Minor, BWV 775)*. [online] available from <https://www.youtube.com/watch?v=J3Hj38wuom0> (©David Magyel) [14 March 2019]

[14] Beethoven, L. van (1810) *Für Elise (Bagatelle No. 25 In A Minor)*. [online] available from <https://www.youtube.com/watch?v=8UJAol7ndfM> (©HSCC MUSIC, Piano Sheet Music) [14 March 2019]

[15] Chopin, F. F. (1829) *Etude Op. 10 No. 9 (F Minor)*. [online] available from <https://www.youtube.com/watch?v=A0umohcLS1I> (©newFranzFerencLiszt) [14 March 2019]

[16] Coldplay (Martin, C., Buckland, J., Berryman, G., Champion, W. and Harvey, P.) (2002) *Clocks*. [online] available from <https://www.youtube.com/watch?v=d020hcWA_Wg> (©Coldplay) [14 March 2019]

[17] Joplin, S. (1899) *Maple Leaf Rag*. [online] available from <https://www.youtube.com/watch?v=_TEbZMfYsLo> (©Paul Barton) [14 March 2019]

[18] Mozart, W. A. (1788) *Piano Sonata No.16 (C Major, KV545)*. [online] available from <https://www.youtube.com/watch?v=dNbqRC4xtEg> (©The Great Repertoire) [14 March 2019]

[19] Mozart, W. A. (1786) *Rondo In D Major (K.485)*. [online] available from <https://www.youtube.com/watch?v=kOcpscSWIno> (©Classical Piano) [14 March 2019]

[20] Pierpont, J. L. (1857) *Jingle Bells*. [online] available from <https://www.youtube.com/watch?v=3PgNPc-iFW8> (©Kids Music) [18 March 2019]

[21] Steiner, M. (1939) *Gone With The Wind (Tara Theme)*. [online] available from <https://www.youtube.com/watch?v=PgF-rcHcPqE> (©EternalLveStory08) [18 March 2019]

[22] Tchaikovsky, P. I. (1876) *Swan Lake (Op. 20, First Movement, Intro)*. [online] available from <https://www.youtube.com/watch?v=surrZsh9rZo> (©HSCC MUSIC, Piano Sheet Music) [18 March 2019]

[23] Pachelbel, J. (1653-1706) *Canon (In D)*. [online] available from <https://www.youtube.com/watch?v=jQDMDQuGrVc> (©HSCC MUSIC, Piano Sheet Music) [18 March 2019]

[24] Akeboshi, Y. (2002) *Wind (Naruto Ed. 01)*. [online] available from <https://www.youtube.com/watch?v=IcseamG7ReY> (©MinecraftN00B67) [18 March 2019]

[25] GOT7 (Tuan, Y. (Mark), Im, J. (JB), Wang, K. (Jackson), Park, J., Choi, Y., Bhuwakul, K. (Bambam) and Kim, Y.) (2015) *Just Right*. [online] available from <https://www.youtube.com/watch?v=vrdk3IGcau8> (©jypentertainment) [18 March 2019]

[26] Hee, M. (1995) *Moonlight In The City (城里的月光)*. [online] available from <https://www.youtube.com/watch?v=zAfmXlv9ff4> (©hknstiu) [18 March 2019]

[27] Mozart, W. A. and Taylor, J. (1806) *Twinkle, Twinkle, Little Star (Twelve Variations On "Ah Vous Dirai-Je, Maman", K.265)*. [online] available from <https://www.youtube.com/watch?v=8A9GJ734nGs> (©Solángel !) and <https://www.youtube.com/watch?v=yCjJyiqpAuU> (©Super Simple Songs - Kids Songs) [18 March 2019]

[28] Oasis (Gallagher, L., Gallagher, N., Arthurs, P. and White, A.) (1995) *Wonderwall*. [online] available from <https://www.youtube.com/watch?v=6hzrDeceEKc> (©Oasis) [18 March 2019]

[29] T-ara (Jeon, B., Lee (Qri), J., Park, S., Hahm, E., Park (Hyomin), S., Park, J. and Lee, A.) (2012) *Sexy Love*. [online] available from <https://www.youtube.com/watch?v=ShVRP09NCO4> (©1theK (원더케이)) [18 March 2019]

[30] Trouble Maker (Jang, H. and Kim, H.) (2011) *Trouble Maker*. [online] available from <https://www.youtube.com/watch?v=s-zRAQmKUkI> (©Trouble Maker (Official YouTube Channel)) [18 March 2019]

[31] Wang, L. (2011) *Still In Love With You (依然爱你)*. [online] available from <https://www.youtube.com/watch?v=sU_ByeHJtw8> (©王力宏 Wang Leehom) [18 March 2019]

[32] Wu, C. (Wu Bai) (2006) *(You Are My) Flower (妳是我的花朵)*. [online] available from <https://www.youtube.com/watch?v=FXg4LXsg14s> (©愛貝克思 avex taiwan) [18 March 2019]

[33] Kwon, J. (G-Dragon) (2009) *Heartbreaker*. [online] available from <https://www.youtube.com/watch?v=LOXEVd-Z7NE> (©YG ENTERTAINMENT) [26 March 2019]

[34] Dillard, T. L. (Flo Rida) feat. Sebert, K. R. (Kesha) (2009) *Right Round*. [online] available from <https://www.youtube.com/watch?v=CcCw1ggftuQ> (©Flo Rida) [26 March 2019]

[35] Kwon, J. (G-Dragon) feat. Dillard, T. L. (Flo Rida) (2010) *Heartbreaker*. [online] available from <https://www.youtube.com/watch?v=1V7AeACaXpk> (©BinladiN EntertainmenT) [26 March 2019]

[36] Kwon, J. (G-Dragon) feat. Jung, J. (Jin Jung) (2009) *Butterfly*. [online] available from <https://www.youtube.com/watch?v=_k0GsfWrNh0> (©YG ENTERTAINMENT) [26 March 2019]

[37] Oasis (Gallagher, L., Gallagher, N., Arthurs, P., McGuigan, P. and White, A.) (1995) *She's Electric*. [online] available from <https://www.youtube.com/watch?v=h9JZWhjQDvc> (©OasisVEVO) [26 March 2019]

[38] Smith, S. (2014) *Stay With Me*. [online] available from <https://www.youtube.com/watch?v=pB-5XG-DbAA> (©SAM SMITH) [26 March 2019]

[39] Petty, T. and The Heartbreakers (1989) *I Won't Back Down*. [online] available from <https://www.youtube.com/watch?v=nvlTJrNJ5lA> (©tompetty) [26 March 2019]

[40] GOT7 (Tuan, Y. (Mark), Im, J. (JB), Wang, K. (Jackson), Park, J., Choi, Y., Bhuwakul, K. (Bambam) and Kim, Y.) (2017) *Never Ever*. [online] available from <https://www.youtube.com/watch?v=IZ1t7CwfvEc> (©jypentertainment) [26 March 2019]

[41] BTS (Kim, S. (Jin), Min, Y. (Suga), Jung, H. (J-Hope), Kim, N. (RM), Park, J., Kim, T. (V) and Jeon, J.) (2017) *Not Today*. [online] available from <https://www.youtube.com/watch?v=9DwzBICPhdM> (©ibighit) [26 March 2019]

[42] KARD (Kim, T. (J.Seph), Kim, M. J. (BM), Jeon, S. and Jeon, J.) (2017) *Don't Recall*. [online] available from <https://www.youtube.com/watch?v=41Dp7Q-SM1Y> (©KARD) [26 March 2019]

[43] Green Day (Armstrong, B., Pritchard, M. (Mike Dirnt), Wright III, F. (Tré Cool)) feat. Cavallo, R. and Freese, J. (2004) *Boulevard Of Broken Dreams*. [online] available from <https://www.youtube.com/watch?v=Soa3gO7tL-c> (©Green Day) [26 March 2019]

[44] Bach, J. S. (1723) *Two-Part Inventions No.1 (C Major, BWV 772)*. [online] available from <https://www.youtube.com/watch?v=8kW27KTLVLo> (©David Magyel) [26 March 2019]

[45] BASTARZ (Block B) (Lee, M., Kim, Y. and Pyo, J.) (2015) *품행제로 (Zero For Conduct)*. [online] available from <https://www.youtube.com/watch?v=-C7vxJ8cB-Y> (©Stone Music Entertainment)

[45a] D.J.S 137 (covers Bastarz of Block B) (2015) *BASTARZ (Block b) - Zero For Conduct 품행제로 (Piano Tutorial) [Sheets + MIDI]*. [online] available from <https://www.youtube.com/watch?v=4TTk-ptKauM> (©D.J.S 137) [26 March 2019]

## References

Anon. (n.d.) *CHAPTER 3 – PROCESS MODELS. - Wikistudent, Unisa Student Wiki* [online] available from <https://studylib.net/doc/8239232/chapter-3-%E2%80%93-process-models.---wikistudent--unisa-student-...> [12 April 2019]

Baidu Music Homepage and Qianqian Music Homepage (2019) *千千音乐-听见世界 (©2015-2019, Merged In 2015)* [online] available from <http://music.taihe.com/> [12 March 2019]

Bello, J. P., Monti, G. and Sandler, M. (2000) "Techniques For Automatic Music Transcription". *Proc. 1St International Symposium On Music Information Retrieval (MUSIC IR 2000)* [online] 1-8. available from <https://ismir2000.ismir.net/> [14 May 2019]

Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H. and Klapuri, A. (2013) "Automatic Music Transcription: Challenges And Future Directions". *Journal Of Intelligent Information Systems* [online] 41 (3), 407-434. available from <http://openaccess.city.ac.uk/2524/1/JIIS-MIRrors-AMT-postprint.pdf> [3 April 2019]

Bhuvaneswari, T. and Prabaharan, S. (2013) "A Survey On Software Development Life Cycle Models". *International Journal Of Computer Science And Mobile Computing* [online] 2 (5), 262 – 267. available from <http://d.researchbib.com/f/2nq3q3YzydL3AgLl5wo20iMT9wpl9jLKOypaZiGJS5ZwNkZl9JZxx1ZwNkZmt0YaOxMt.pdf> [12 April 2019]

Boisvert, R., Moreira, J., Philippsen, M. and Pozo, R. (2001) "Java And Numerical Computing". *Computing In Science & Engineering* [online] 3 (2), 18-24. available from <https://ieeexplore.ieee.org/abstract/document/908997> [23 April 2019]

Brandenburg, K. (1999) "MP3 And AAC Explained." in *The Proceedings Of the 17Th International Conference: High-Quality Audio Coding (September 1999)* [online] held 1999. Audio Engineering Society, 17-009. available from <http://www.aes.org/e-lib/browse.cfm?elib=8079> [24 January 2019]

Briot, J., Hadjeres, G. and Pachet, F. (2017) "Deep Learning Techniques For Music Generation-A Survey". *arXiv Preprint, arXiv:1709.01620* [online] available from <https://arxiv.org/abs/1709.01620> [22 September 2019]

Cano, P., Batlle, E., Gómez, E., de C.T.Gomes, L. and Bonnet, M. (2005) "Audio Fingerprinting: Concepts And Applications". *Computational Intelligence For Modelling And Prediction* [online] 2, 233-245. available from <https://link.springer.com/chapter/10.1007/10966518_17#citeas> [27 February 2019]

Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C. and Slaney, M. (2008) "Content-Based Music Information Retrieval: Current Directions And Future Challenges". *Proceedings Of The IEEE*[online] 96 (4), 668-696. available from <https://ieeexplore.ieee.org/abstract/document/4472077> [27 February 2019]

Chen, C., Jang, J., Liu, W. and Weng, C. (2016) "An Efficient Method For Polyphonic Audio-To-Score Alignment Using Onset Detection And Constant Q Transform". *2016 IEEE International Conference On Acoustics, Speech And Signal Processing (ICASSP)* [online] 2802-2806. available from <https://ieeexplore.ieee.org/abstract/document/7472188> [28 March 2019]

Chen, X. and Lin, X. (2014) "Big Data Deep Learning: Challenges And Perspectives". *IEEE Access* [online] 2, 514-525. available from

<https://s3.amazonaws.com/academia.edu.documents/37708688/BigDataDeepLearningChallengesand.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1549387877&Signature=Wt5cVH4SHHH0cgMVNR62Mv2C%2BPw%3D&response-content-disposition=inline%3B%20filename%3DBig_Data_Deep_Learning_Challenges_and_Pe.pdf> [5 February 2019]

Clarke, E. (2004) "Empirical Methods In The Study Of Performance." in *Empirical Musicology: Aims, Methods, Prospects* [online] ed. by Clarke, E. and Cook, N. New York, USA: Oxford University Press, 77-102. available from <https://s3.amazonaws.com/academia.edu.documents/31772787/58264440-Clarke-Cook-Empirical-Musicology-Aims-Methods-Prospects-Oxford-2004-1.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1556291577&Signature=utkYY%2BIjopw2fOLOXax4x4NX%2FGs%3D&response-content-disposition=inline%3B%20filename%3DEmpirical_Musicology_-_Aims_Methods_Pros.pdf#page=86> [26 April 2019]

Coelho, M. and Zigelbaum, J. (2011) "Shape-Changing Interfaces". *Personal And Ubiquitous Computing* [online] 15 (2), 161-173. available from <https://dl.acm.org/citation.cfm?id=1938295> [6 February 2019]

Cogliati, A., Temperley, D. and Duan, Z. (2016) "Transcribing Human Piano Performances Into Music Notation". In *Proc. 17Th International Conference On Music Information Retrieval* [online] 758-764. available from <http://m.mr-pc.org/ismir16/website/articles/088_Paper.pdf> [19 June 2019]

Damm, B. (1993) *The Development Of A Piano-Recorder System*. Ph.D. Cape Technikon

Dannenberg, R.B. (1993) "Music Representation Issues, Techniques, And Systems". *Computer Music Journal* [online] 17 (3), 20. available from <https://www.jstor.org/stable/3680940?seq=1#metadata_info_tab_contents> [19 June 2019]

Demopoulos, R. and Katchabaw, M. (2007) *Music Information Retrieval: A Survey Of Issues And Approaches* [online] Technical Report #677. Ontario, Canada: The University of Western Ontario. available from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.3353&rep=rep1&type=pdf> [12 March 2019]

Derrien, O. (2014) "A Very Low Latency Pitch Tracker For Audio To MIDI Conversion". *Proc. Of 17Th International Conference On Digital Audio Effects (Dafx-14)* [online] 1-6. available from <https://pdfs.semanticscholar.org/225a/2660887ebb9b62f3892429c65b95806325f0.pdf> [19 June 2019]

Dittmar, C., Hildebrand, K., Gaertner, D., Winges, M., Müller, F. and Aichroth, P. (2012) "Audio Forensics Meets Music Information Retrieval — A Toolbox For Inspection Of Music Plagiarism." in *Proceedings Of The 20Th European Signal Processing Conference (EUSIPCO)* [online] held 2012. IEEE, 1249-1253. available from <https://ieeexplore.ieee.org/abstract/document/6333813> [30 January 2019]

Dixon, S. (2000) "On The Computer Recognition Of Solo Piano Music". *Proceedings Of Australasian Computer Music Conference* [online] 31-37. available from <http://www.eecs.qmul.ac.uk/~simond/pub/2000/acmc.pdf> [13 February 2019]

Domingos, P. (2012) "A Few Useful Things To Know About Machine Learning". *Communications Of The ACM* [online] 55 (10), 78-87. available from <https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf> [10 February 2020]

Duong, N. and Duong, H. (2015) "A Review Of Audio Features And Statistical Models Exploited For Voice Pattern Design". *Seventh International Conferences On Pervasive Patterns And Applications (PATTERNS 2015, Nice, France)* [online] available from <https://arxiv.org/abs/1502.06811> [4 March 2019]

Elmasri, R. and Navathe, S. (2014) *Fundamentals Of Database Systems.* 6th edn. (Pearson New International Edition) Boston: Addison wesley

Esther. (2013) *Korean Boyband BIGBANG Cited for Plagiarism in High School Textbook* [online] available from < https://soranews24.com/2013/02/27/korean-boyband-bigbang-cited-for-plagiarism-in-high-school-textbook/> [30 January 2020]

Evans, R., Pfahringer, B. and Holmes, G. (2011) "Clustering For Classification". *2011 7Th International Conference On Information Technology In Asia* 1-8

Fan, Y. and Feng, S. (2016) "A Music Identification System Based On Audio Fingerprint". *2016 4Th Intl Conf On Applied Computing And Information Technology/3Rd Intl Conf On Computational Science/Intelligence And Applied Informatics/1St Intl Conf On Big Data, Cloud Computing, Data Science & Engineering (ACIT-CSII-BCD)* [online] 363-367. available from <https://ieeexplore.ieee.org/abstract/document/7917011> [13 March 2019

Farber, M. (2003) "Topological Complexity Of Motion Planning". *Discrete And Computational Geometry* [online] 29 (2), 211-221. available from <https://link.springer.com/content/pdf/10.1007%2Fs00454-002-0760-9.pdf> [6 February 2019]

Fournier-S'niehotta, R., Rigaux, P. and Travers, N. (2016) "Is There A Data Model In Music Notation?". In *Intl. Conf. On Technologies For Music Notation And Representation (TENOR'16)* [online] 85-91. available from <http://tenor2016.tenor-conference.org/papers/12_Fournier_tenor2016.pdf> [19 June 2019]

Francisco, M. (2013-2019) *Miggy Smallz Youtube Homepage* [online] available from <https://www.youtube.com/channel/UC5XWNylwy4efFufjMYqcglw> [6 March 2019]

Francisco, M. (Miggy Smallz) (2017) *GOT7 X BTS X K.A.R.D - Never Ever / Not Today / Don't Recall MASHUP* [online] available from <https://www.youtube.com/watch?v=Xz__aAyOAB4> [26 March 2019]

Fremerey, C., Müller, M., Kurth, F. and Clausen, M. (2008) "Automatic Mapping Of Scanned Sheet Music To Audio Recordings". In *Proc. 9Th International Conference On Music Information Retrieval* [online] 413-418. available from <http://ismir2008.ismir.net/papers/ISMIR2008_116.pdf> [19 June 2019]

Gaon Chart (2010.03.28-2010.04.03) (2010) *국내 대표 음악 차트 가온차트! (Korean Representative Music Chart Gaon Chart!)* [online] available from <http://gaonchart.co.kr/main/section/chart/online.gaon?nationGbn=T&serviceGbn=ALL&targetTime=15&hitYear=2010&termGbn=week> [31 January 2019]

Gerou, T. and Lusk, L. (1996) *Essential Dictionary Of Music Notation*. Los Angeles, CA: Alfred Pub.

Gomersall, V. and Clarke, H. (n.d.) *Tune Or Melody Lookup - Dictionary Of Musical Themes*[online] available from <http://bestclassicaltunes.com/DictionaryPiano.aspx> [13 March 2019]

Good, M. (2001) "Musicxml For Notation And Analysis." in *The Virtual Score: Representation, Retrieval, Restoration* [online] ed. by Hewlett, W. and Selfridge-Field, E. Cambridge:

MIT Press, 113-124. available from <http://www.ccarh.org/publications/books/cm/vol/12/> [28 January 2019]

Good, M. and Mills, S. (2015) *Distribution Of Audio Sheet Music As An Electronic Book*. US 8,933,312 B2. U.S.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014) "Generative Adversarial Nets". *Advances In Neural Information Processing Systems* 2672-2680

Guillaume, P. (2006) *Music And Acoustics*. London: ISTE

Haitsma, J. and Kalker, T. (2002) "A Highly Robust Audio Fingerprinting System." in *Proc. 3Rd International Conference On Music Information Retrieval* [online] held 2002 at Paris, France. 107-115. available from <http://ismir2002.ircam.fr/proceedings/02-FP04-2.pdf> [4 March 2019]

Haitsma, J. and Kalker, T. (2003) "A Highly Robust Audio Fingerprinting System With An Efficient Search Strategy". *Journal Of New Music Research* [online] 32 (2), 211-221. available from <https://www.tandfonline.com/doi/abs/10.1076/jnmr.32.2.211.16746> [4 March 2019]

Han, S. (2009) *Big Bang Leader Accused Of Plagiarism* [online] available from <http://m.koreatimes.co.kr/pages/article.asp?newsIdx=50616> [30 January 2019]

Hatcher, A. (2002) *Algebraic Topology*. New York: Cambridge University Press

Heaton, J. (2015) *Artificial Intelligence For Humans, Volume 3: Deep Learning And Neural Networks*. St. Louis, MO, USA: Heaton Research, Inc.

Holm, J., Havukainen, K. and Arrasvuori, J. (2005) "Personalizing Game Content Using Audio-Visual Media". *ACM International Conference Proceeding Series* [online] 265, 298-301. available from <https://www.researchgate.net/profile/Juha_Arrasvuori/publication/220982728_Personalizing_game_content_using_audio-visual_media/links/543ce7f10cf20af5cfbf870b.pdf> [2 April 2019]

Hong, L. and Cai, J. (2010) "The Application Guide Of Mixed Programming Between MATLAB And Other Programming Languages". *2010 The 2Nd International Conference On Computer And Automation Engineering (ICCAE)* [online] 185-189. available from <https://ieeexplore.ieee.org/abstract/document/5452058> [18 April 2019]

Hrušková, N. and Hvolka, J. (2011) "Representing, Comparing And Evaluating Of Music Files." in *The Proceedings Of the International Conference On E-Learning And The Knowledge Society* [online] held 2011. ASE Publishing House, 978-606. available from <https://pdfs.semanticscholar.org/2c62/24ada67e794a2ca37b12067c266ebd9a9d09.pdf> [29 January 2019]

Hu, X. (2018) "Evaluating Mobile Music Services In China: An Exploration In User Experience". *Journal Of Information Science* [online] 45 (1), 16-28. available from <https://journals.sagepub.com/doi/pdf/10.1177/0165551518762070> [22 January 2019]

Huang, T., Xia, G., Ma, Y., Dannenberg, R. and Faloutsos, C. (2013) "Midifind: Fast And Effective Similarity Searching In Large MIDI Databases". *Proc. Of The 10Th International Symposium On Computer Music Multidisciplinary Research* [online] 209-224. available from <http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rbd/www/papers/MidiFind-CMMR-2013.pdf> [20 March 2019]

Huron, D., Sapp, C., Kornstädt, A., van Handel, L., Chafe, Z., Heifitz, M., Leistikow, K., Hewlett, W., Selfridge-Field, E., Wiering, F. and Aarden, B. (n.d.) *Themefinder* [online] available from <http://www.themefinder.org/> [13 March 2019]

Joutsenvirta, A. and Perkiömäki, J. (2010) *Music Theory 1* [online] available from <http://www2.siba.fi/muste1/index.php?id=2&la=en> [14 May 2019]

Kania, A. (2017) "The Philosophy Of Music." in *The Stanford Encyclopedia Of Philosophy (Fall 2017 Edition)* [online] ed. by Zalta, E. Metaphysics Research Lab, Stanford University. available from <https://plato.stanford.edu/archives/spr2014/entries/music/#toc> [17 January 2019]

Kessler, B. (2007) "Und" Approach To Fourier Transforms: Using Music To Teach Trigonometry". *Mathematics Faculty Publications & 2007 Bridges Donostia Conference* [online] 135-142. available from <https://digitalcommons.wku.edu/cgi/viewcontent.cgi?referer=https://scholar.google.co.uk/&httpsredir=1&article=1012&context=math_fac_pub> [30 May 2019]

Kim, S., Kim, J. and Shin, P. (2013) "Designing A Music Content Retrieval System Supporting Korean". *2013 International Conference On Information Science And Applications (ICISA)* [online] 1-4. available from <https://ieeexplore.ieee.org/abstract/document/6579492> [9 April 2019]

Klapuri, A. and Davy, M. (2006) *Signal Processing Methods For Music Transcription*. New York: Springer

Knees, P. and Schedl, M. (2013) "A Survey Of Music Similarity And Recommendation From Music Context Data". ACM Transactions On Multimedia Computing, Communications, And Applications[online] 10 (1), 1-21. available from <https://dl.acm.org/citation.cfm?id=2542206> [30 January 2019]

Kohonen, T. (1990) "The Self-Organizing Map". *Proceedings Of The IEEE* 78 (9), 1464-1480

Kugou Homepage (2019) *酷狗音乐 - 就是歌多 (©2004-2019)* [online] available from <http://www.kugou.com/> [5 March 2019]

Kuwo Homepage (2019) *酷我音乐_无损音乐正版在线试听网站 (©2005-2019)* [online] available from <http://www.kuwo.cn/> [12 March 2019]

Kwon, T., Jeong, D. and Nam, J. (2017) "Audio-To-Score Alignment Of Piano Music Using RNN-Based Automatic Music Transcription". *Proceedings Of The 14Th Sound And Music Computing Conference* [online] (SMC2017-)380-385. available from <https://arxiv.org/abs/1711.04480> [16 April 2019]

Lamere, P. (2008) "Social Tagging And Music Information Retrieval". *Journal Of New Music Research* [online] 37 (2), 101-114. available from <https://www.tandfonline.com/doi/abs/10.1080/09298210802479284> [27 February 2019]

LaValle, S. (2006) "The Configuration Space." in *Planning Algorithms* [online] ed. by LaValle, S. Cambridge university press, 127-184. available from <http://planning.cs.uiuc.edu/node1.html> [6 February 2019]

Lenssen, N. and Needell, D. (2014) "An Introduction To Fourier Analysis With Applications To Music". *Journal Of Humanistic Mathematics* [online] 4 (1), 72-91. available from <https://www.researchgate.net/profile/Nathan_Lenssen/publication/314918556_An_Introduction_to_Fourier_Analysis_with_Applications_to_Music/links/595d431c0f7e9b3aefa

df009/An-Introduction-to-Fourier-Analysis-with-Applications-to-Music.pdf> [13 February 2019]

Liao, C., Wang, P. and Zhang, Y. (2009) "Mining Association Patterns Between Music And Video Clips In Professional MTV". *Lecture Notes In Computer Science* [online] 5371, 401-412. available from <https://link.springer.com/chapter/10.1007/978-3-540-92892-8_41> [9 April 2019]

Liu, B., Xia, Y. and Yu, P. (2000) "Clustering Through Decision Tree Construction". *Proceedings Of The Ninth International Conference On Information And Knowledge Management* [online] 20-29. available from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.4310&rep=rep1&type=pdf> [26 February 2019]

Loy, G. (1985) "Musicians Make A Standard: The MIDI Phenomenon". *Computer Music Journal* [online] 9 (4), 8-26. available from <https://www.jstor.org/stable/pdf/3679619.pdf> [4 March 2019]

Lubiw, A. and Tanur, L. (2004) "Pattern Matching In Polyphonic Music As A Weighted Geometric Translation Problem." in *Proc. 5Th International Conference On Music Information Retrieval* [online] held 2004 at Barcelona, Spain. 289--296. available from <http://ismir2004.ismir.net/proceedings/p054-page-289-paper154.pdf> [4 March 2019]

Matityaho, B. and Furst, M. (1995) "Neural Network Based Model For Classification Of Music Type". *Eighteenth Convention Of Electrical And Electronics Engineers In Israel* [online] 4.3.4.1-5. available from <https://ieeexplore.ieee.org/abstract/document/514161> [9 April 2019]

MATLAB and Simulink Homepage (2019) *Mathworks - Makers Of MATLAB And Simulink (©1994-2019)* [online] available from <https://uk.mathworks.com/> [24 April 2019]

McKay, C. and Fujinaga, I. (2006) "Jsymbolic: A Feature Extractor For MIDI Files". *Proceedings Of International Computer Music Conference* [online] 302-305. available from <https://www.semanticscholar.org/paper/jSymbolic%3A-A-Feature-Extractor-for-MIDI-Files-McKay-Fujinaga/9d3ca8a8b5c5bc0e7bc7d22d13b83f8c60f155cf> [4 March 2019]

McKay, C., Tenaglia, T. and Fujinaga, I. (2016) *Jsymbolic2: Extracting Features From Symbolic Music Representations* [online] Late-Breaking Demo Session of the 17Th International Conference On Music Information Retrieval. New York City, USA: ISMIR. available from <https://pdfs.semanticscholar.org/e67e/2091d5b2f9bcecfb89940b9d725ecc022098.pdf > [20 March 2019]

Mesaros, A. and Virtanen, T. (2008) "Automatic Alignment Of Music Audio And Lyrics." in *Proceedings Of The 11Th Int. Conference On Digital Audio Effects (Dafx-08)* [online] held 2008. available from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.212.6683&rep=rep1&type=pdf> [29 January 2019]

Miljkovic, D. (2017) "Brief Review Of Self-Organizing Maps". *2017 40Th International Convention On Information And Communication Technology, Electronics And Microelectronics (MIPRO)* [online] available from <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7973581> [12 February 2019]

Müllensiefen, D. and Wiggins, G. (2011) "Polynomial Functions As A Representation Of Melodic Phrase Contour." [online] in *Systematic Musicology: Empirical And Theoretical Studies,* Vol 28 *Of Hamburger Jahrbuch Für Musikwissenschaft*. ed. by Schneider, A. and von Ruschkowski, A. Peter Lang, 63-88. available from

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.410.3479&rep=rep1&type=pdf> [20 June 2019]

Musicradar Homepage (2019) *Musicradar: The No.1 Website For Musicians (©2007-2019)* [online] available from <https://www.musicradar.com/> [12 March 2019]

Netease Cloud Music Homepage (2019) *网易云音乐 (©1997-2019)* [online] available from <https://st.music.163.com/c/gdpr/index.html> [12 March 2019]

OasisIndie94 (2011) *Green Day Feat. Oasis - Boulevard Of Broken Dreams & Wonderwall* [online] available from <https://www.youtube.com/watch?v=z-PdV61Z8Mg> [6 March 2019]

Osmalsky, J., Embrechts, J., Van Droogenbroeck, M. and Pierard, S. (2012) "Neural Networks For Musical Chords Recognition". *Journées D'informatique Musicale* [online] 39-46. available from <http://hdl.handle.net/2268/115963> [9 April 2019]

Park, J., Kim, S. and Shin, M. (2005) "Music Plagiarism Detection Using Melody Databases". *Lecture Notes In Computer Science* [online] 684-693. available from <https://link.springer.com/chapter/10.1007/11553939_98> [30 January 2019]

Park, S.B., Kwon, T.G., Lee, J.P., Kim, J.H. and Nam, J.H., 2019. A Cross-scape Plot Representaion for Visulizing Symbolic Melodic Similarity. In *Proc. 20th International Conference on Music Information Retrieval*. 4-8 Nov 2019, Delft, Netherland. available from < http://archives.ismir.net/ismir2019/paper/000050.pdf > [23 March 2020]

Pascual, S., Bonafonte, A. and Serra, J. (2017) "SEGAN: Speech Enhancement Generative Adversarial Network". *Arxiv Preprint Arxiv:1703.09452*

Polikar, R. (2006) "Ensemble Based Systems In Decision Making". *IEEE Circuits And Systems Magazine* [online] 6 (3), 21-45. available from <https://ieeexplore.ieee.org/document/1688199> [26 February 2019]

Portnoff, M. (1976) "Implementation Of The Digital Phase Vocoder Using The Fast Fourier Transform". *IEEE Transactions On Acoustics, Speech, And Signal Processing* [online] 24 (3), 243-248. available from <https://pdfs.semanticscholar.org/4d37/0876b843d5c8fcca151e548939b8744ce8bc.pdf> [13 February 2019]

Prechelt, L. (2003) "Are Scripting Languages Any Good? A Validation Of Perl, Python, Rexx, And Tcl Against C, C++, And Java." in *Advances In Computers: Information Repositories (Vol. 57)* [online] 1st edn. ed. by Zelkowitz, M. Great Britain: Academic Press, 205-270. available from <http://www.inf.fu-berlin.de/inst/ag-se/pubs/jccpprt2_advances2003.pdf> [24 April 2019

Priemer, R. (1991) *Introductory Signal Processing*. Singapore: World Scientific

QQ Music Homepage (2019) *QQ 音乐-千万正版音乐海量无损曲库新歌热歌天天畅听的高品质音乐平台！(©1998-2019)* [online] available from <https://y.qq.com/> [12 March 2019]

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B. and Lee, H. (2016) "Generative Adversarial Text To Image Synthesis". *Arxiv Preprint Arxiv:1605.05396*

Rothstein, J. (1995) *Midi: A Comprehensive Introduction (Vol. 7)*. 2nd edn. Madison, WI: A-R Editions, Inc

Rowe, R. (2009) "Split Levels: Symbolic To Sub-Symbolic Interactive Music Systems". *Contemporary Music Review* [online] 28 (1), 31-42. available from

<https://www.tandfonline.com/doi/full/10.1080/07494460802664015?scroll=top&needAccess=true> [24 January 2019]

Safavian, S. and Landgrebe, D. (1991) "A Survey Of Decision Tree Classifier Methodology". *IEEE Transactions On Systems, Man, And Cybernetics* [online] 21 (3), 660-674. available from <https://ieeexplore.ieee.org/abstract/document/97458> [26 February 2019]

Saxena, A., Jasola, S., Barik, K., Joshi, A., Dubey, A. and Jauhari, M. (2018) "Unit-9: The Computer System: Hardware For Educational Computing." in *Block-3: Introduction To Computers In Education* [online] IGNOU, 5-18. available from <http://www.egyankosh.ac.in/bitstream/123456789/47080/1/Unit-9.pdf> [26 April 2019]

Schmidhuber, J. (2015) "Deep Learning In Neural Networks: An Overview". *Neural Networks* [online] 61, 85-117. available from <https://www.sciencedirect.com/science/article/pii/S0893608014002135> [5 February 2019]

Sharma, N. and Sardana, S. (2016) "Designing A Real Time Speech Recognition System Using MATLAB". *International Journal Of Computer Applications (0975 – 8887) National Conference On Latest Initiatives& Innovations In Communication And Electronics (Lice 2016)* [online] 1-5. available from <https://pdfs.semanticscholar.org/ea23/5ca13b4884c5d0b00a65e3782faed76a6883.pdf> [13 February 2019]

Shazam Homepage (2019) *Shazam (©1999-2019)* [online] available from <https://www.shazam.com/> [5 March 2019]

Shin, M., Kim, E., Seo, S. and Kim, Y. (2016) "Similarity Measurement System Design Of Music Contents". *International Information Institute (Tokyo). Information* [online] 19 (1), 67-72. available from <https://search.proquest.com/openview/9f866b5258c2a7be4e7b4d9c7449452e/1?pq-origsite=gscholar&cbl=936334> [30 January 2019]

Song, Y., Dixon, S. and Marcus, P. (2012) "A Survey Of Music Recommendation Systems And Future Perspectives." in 9Th International Symposium On Computer Music Modelling And Retrieval (CMMR 2012) [online] held 2012 at London, United Kingdom. available from <https://www.researchgate.net/profile/Yading_Song/publication/277714802_A_Survey_of_Music_Recommendation_Systems_and_Future_Perspectives/links/5571726608aef8e8dc633517.pdf> [30 January 2019]

SoundHound Homepage (2019) *Soundhound Inc. (©2005-2019)* [online] available from <https://soundhound.com/> [12 March 2019]

Spanias, A., Painter, T. and Atti, V. (2006) *Audio Signal Processing And Coding*. Canada: John Wiley & Sons

Sri Ranjani, S., Abdulkareem, V., Karthik, K. and Bora, P. (2015) "Application Of SHAZAM-Based Audio Fingerprinting For Multilingual Indian Song Retrieval". *Lecture Notes In Electrical Engineering*[online] 81-92. available from <https://link.springer.com/chapter/10.1007/978-81-322-2464-8_6> [4 March 2019]

Sumanac-Johnson, D. (2016) *You Stole My Song! 4 Historical Examples Of Music Plagiarism | CBC News* [online] available from <https://www.cbc.ca/news/arts/music-plagiarism-analysis-1.3634429> [17 January 2019]

Sutton, R., Barto, A. and Bach, F. (2018) *Reinforcement Learning: An Introduction*. 2nd edn. MIT Press

Tan, L. and Jiang, J. (2018) *Digital Signal Processing: Fundamentals And Applications*. 3rd edn. Academic Press

Tan, P., Steinbach, M., Karpatne, A. and Kumar, V. (2018) *Introduction To Data Mining (Second Edition)*. 2nd edn. India: Pearson Education [online] available from < https://www-users.cs.umn.edu/~kumar001/dmbook/index.php> [26 February 2019]

Tang, Z. (2016) *Study And Evaluation Of Current Music Information Retrieval System*. Undergraduate. Coventry University

Tolstov, G. (2009) *Fourier Series*. New York: Dover Publ.

TrackID Homepage (2017) *Trackid™ - Music Discovery, Recognize The Music Around You (©2015-2017)* [online] available from <https://trackid.sonymobile.com/> [before 15 September 2017]

Typke, R. (2019) *Musipedia: Musipedia Melody Search Engine* [online] available from <https://www.musipedia.org/> [5 March 2019]

Typke, R., Wiering, F. and Veltkamp, R. (2005) "A Survey Of Music Information Retrieval Systems." in *Proc. 6Th International Conference On Music Information Retrieval* [online] held 2005 at Queen Mary, University of London. 153-160. available from <https://www.researchgate.net/profile/Remco_Veltkamp/publication/220723096_A_Survey_of_Music_Information_Retrieval_Systems/links/0deec51df16a5d53e6000000/A-Survey-of-Music-Information-Retrieval-Systems.pdf> [30 January 2019]

Valero, D. and Quereda, J. (2010) *Symbolic Music Comparison With Tree Data Structures*. Ph.D. Universidad de Alicante

Viitaniemi, T., Klapuri, A. and Eronen, A. (2003) "A Probabilistic Model For The Transcription Of Single-Voice Melodies". *Proceedings Of The 2003 Finnish Signal Processing Symposium (FINSIG'03)* [online] 59-63. available from <http://www.ee.columbia.edu/~dpwe/papers/ViitKE03-melodies.pdf> [14 May 2019]

Walder, C. (2016) "Modelling Symbolic Music: Beyond The Piano Roll." in *Proc. Of The 8Th Asian Conference On Machine Learning (ACML)* [online] held 2016 at Hamilton, New Zealand. JMLR: Workshop and Conference Proceedings, 174-189. available from <http://proceedings.mlr.press/v63/walder88.pdf> [4 March 2019]

Wang, A. L. (2003) "An Industrial Strength Audio Search Algorithm". In *Proc. 4Th International Conference On Music Information Retrieval* [online] 7-13. available from <http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf> [5 July 2019]

Widrow, B. and Lehr, M.A. (1990) "30 Years Of Adaptive Neural Networks: Perceptron, Madaline, And Backpropagation". *Proceedings Of The IEEE* [online] 78 (9), 1415-1442. available from <http://isl-www.stanford.edu/~widrow/papers/j199030years.pdf> [1 July 2019]

Wiering, M. and van Hasselt, H. (2008) "Ensemble Algorithms In Reinforcement Learning". *IEEE Transactions On Systems, Man, And Cybernetics, Part B (Cybernetics)* [online] 38 (4), 930-936. available from <https://ieeexplore.ieee.org/document/4509588> [26 February 2019]

Xue, W. (2018) *Implementation Of A Player For MDL & MML Musical Pieces*. Undergraduate. Coventry University

Yang, N., Usman, M., He, X., Jan, M. and Zhang, L. (2017) "Time-Frequency Filter Bank: A Simple Approach For Audio And Music Separation". *IEEE Access* [online] 5, 27114-27125. available from <https://ieeexplore.ieee.org/abstract/document/8063868> [13 February 2019]

Yu, W., Reid, D. and Brewster, S. (2002) "Web-Based Multimodal Graphs For Visually Impaired People *Universal Access And Assistive Technology* [online] 97-108. available from <https://link.springer.com/chapter/10.1007/978-1-4471-3719-1_10#citeas> [28 January 2019]

Zainuddin Lubis, M., Munandar Manik, H. and Wulandari, P. (2016) *Signal Processing For Marine Acoustic And Dolphin Using Matlab*. LAP LAMBERT Academic Publishing

Zhang, C. and Ma, Y. (2012) *Ensemble Machine Learning*. New York: Springer

# Appendix

## Appendix A. Detailed Project Methodologies

Our project starts with relevant primary and secondary research, the results have shown in Chapter 2. Next, we analysed some typical software development life cycle models (Bhuvaneswari & Prabaharan, 2013; Anon., n.d.), to decide how the rest of the project should proceed. Here are the definitions of the four different types of the development models:

*Definition 11. The Linear Model.*

*The Linear Model is also known as the Waterfall Model. This classic model requires the developer to work on a very organized schedule, the plan is in a sequential order and the developer completes the individual tasks one by one. (e.g., define-before-design and design-before-code). The complete solution only appears at the end of the process.*

*Definition 12. The Iterative Model.*

*Unlike the linear model, this model does not require with the full specification of requirements. During each iteration, we followed the process from the linear model. Further requirement could be identified. By repeating this process, we can get different version of the final product.*

*Definition 13. The Incremental Model.*

*This model involves several developing stages, and the final product is produced after finishing all the increments. We can see the partial core product after one increment stage, and after every increment stage, new function can be added to satisfy user's requirement.*

*Definition 14. The Evolutionary Model.*

*Deal with complex systems, normally a comination of iterative and incrementl model. Typical examples would be the Prototyping, Spiral models.*

Based on (Bhuvaneswari & Prabaharan, 2013), Table A-1 summarizes the pros and cons for those models. According to Section 1.3, our project involves certain general aims and objectives. Moreover, it is not a large project, and have a three years time limitation. Therefore, the prototyping model is a reasonable choice for our project.

The next decision is the choice for the programming language through out the project. Table A-2 shows the pros and cons of several typical programming languages (Boisvert, et al., 2001; Prechelt, 2003; Hong & Cai, 2010). According to Section 1.3, our project is majorly to evaluate music similarity scores by converting music data into mathematical

| | Pros | Cons |
|---|---|---|
| **Linear (Waterfall)** | • Easy to understand and implement.<br>• Fault fixed before carrying out any implementation.<br>• Possible to estimate the development and the time cost. | • Hard to get accurate requirements.<br>• Errors may be discovered at later stages.<br>• Expensive to make any alternations.<br>• Difficult for risk management integration. |
| **Iterative** | • Improve the product step by step.<br>• Possible to get reliable and useful user feedback.<br>• Less documentation time and more design time. | • No overlap for each iteration.<br>• May not get full requirements, and thus, inaccurate design of system architectures. |
| **Incremental** | • Problem can be detected earlier and able to made changes after each cycle.<br>• Fast core product production. | • Heavy documentations.<br>• Difficult for partitioning the functions and features<br>• Long time period. |
| **Evolutionary: Prototyping (leads to Rapid Application Developement)** | • Quick plan and quick design.<br>• Reduces the development time. So faster production.<br>• Components can be reused. | • Fuzzy requirements.<br>• Inefficiency algorithm.<br>• Fail for large project with lacked commitment or poor project modularization. |
| **Evolutionary: Spiral** | • High amount of risk analysis.<br>• Adaptive and allow alteration for funcionalities and requirements.<br>• More completed version of the software can be produced. | • Need to consider all major risks at all stages.<br>• Can be a costly model to use.<br>• Does not work well for small projects. |

*Table A-1. Pros and Cons for the Development Models (Bhuvaneswari & Prabaharan, 2013).*

| | Pros | Cons |
|---|---|---|
| **C/C++/ Java** | • C allows to implement extreme efficient programs (Compiled).<br>• C++ is somehow super set of C, by adding object-oriented features, extensions etc.<br>• Java, resembles C, is object oriented (with important features), and has a higher portability and safety. It allows incorporate into documents describing the behaviour of the architecture. | • C/C++ reliant on well-written programs.<br>• C/C++ are not machine independent and lack of certain high-level language features.<br>• Java needs longer simulation run times.<br>• Java has difficulty for certain coordination components of a large numerical application, e.g., complex number and true multidimensional arrays. |
| **MATLAB** | • Simple and useful with friendly working platform.<br>• Powerful capability of processing mathematical and scientific computer data.<br>• Exellent function of processing graphics.<br>• Widely used toolkit modules. | • Low operating efficiency.<br>• Non cross-platform, low portability.<br>• Relatively poor capability of access to hardware.<br>• Graphical User Interface functions were not flexible enough. |
| **Python** | • Resembles Java with many built-in functions. No special rule when combining them.<br>• Easier for programmer to read and study. | • Slow speed of execution as Python is interpreted, instead of using compilers.<br>• Design restrictions and may raise run-time errors. |

*Table A-2. Pros and Cons for C, C++, Java, MATLAB and Python.*

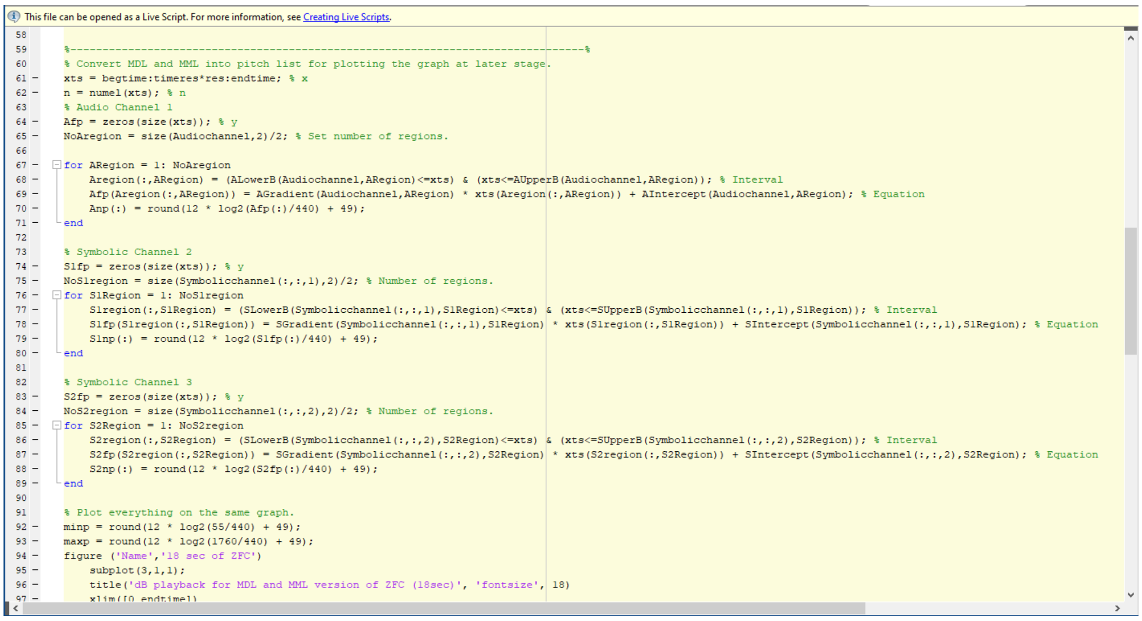
data. I.e., more theoretical than application practical. Thus, from the summary table, MATLAB is the reasonble programming language choice for our project. In addition, it is possible to interface with other programs written in C, C++, Java, Python etc. (MATLAB and Simulink Homepage, 2019).

## Appendix B. Codes

Certain functions and important steps from those Algorithms and Equations were shown in here:

### Algorithm 1

Line 5b:

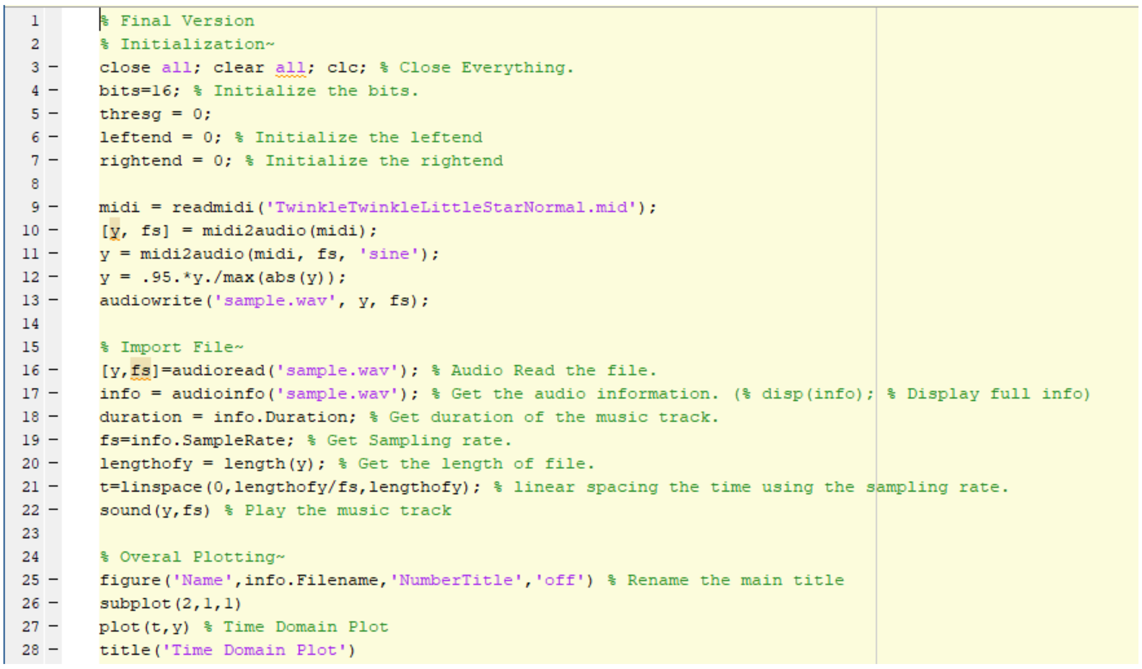


This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```matlab

%-----------------------------------------------------------------------------%
% Convert MDL and MML into pitch list for plotting the graph at later stage.
xts = begtime:timeres*res:endtime; % x
n = numel(xts); % n
% Audio Channel 1
Afp = zeros(size(xts)); % y
NoAregion = size(Audiochannel,2)/2; % Set number of regions.

for ARegion = 1: NoAregion
    Aregion(:,ARegion) = (ALowerB(Audiochannel,ARegion)<=xts) & (xts<=AUpperB(Audiochannel,ARegion)); % Interval
    Afp(Aregion(:,ARegion)) = AGradient(Audiochannel,ARegion) * xts(Aregion(:,ARegion)) + AIntercept(Audiochannel,ARegion); % Equation
    Anp(:) = round(12 * log2(Afp(:)/440) + 49);
end

% Symbolic Channel 2
S1fp = zeros(size(xts)); % y
NoS1region = size(Symbolicchannel(:,:,1),2)/2; % Number of regions.
for S1Region = 1: NoS1region
    S1region(:,S1Region) = (SLowerB(Symbolicchannel(:,:,1),S1Region)<=xts) & (xts<=SUpperB(Symbolicchannel(:,:,1),S1Region)); % Interval
    S1fp(S1region(:,S1Region)) = SGradient(Symbolicchannel(:,:,1),S1Region) * xts(S1region(:,S1Region)) + SIntercept(Symbolicchannel(:,:,1),S1Region); % Equation
    S1np(:) = round(12 * log2(S1fp(:)/440) + 49);
end

% Symbolic Channel 3
S2fp = zeros(size(xts)); % y
NoS2region = size(Symbolicchannel(:,:,2),2)/2; % Number of regions.
for S2Region = 1: NoS2region
    S2region(:,S2Region) = (SLowerB(Symbolicchannel(:,:,2),S2Region)<=xts) & (xts<=SUpperB(Symbolicchannel(:,:,2),S2Region)); % Interval
    S2fp(S2region(:,S2Region)) = SGradient(Symbolicchannel(:,:,2),S2Region) * xts(S2region(:,S2Region)) + SIntercept(Symbolicchannel(:,:,2),S2Region); % Equation
    S2np(:) = round(12 * log2(S2fp(:)/440) + 49);
end

% Plot everything on the same graph.
minp = round(12 * log2(55/440) + 49);
maxp = round(12 * log2(1760/440) + 49);
figure ('Name','18 sec of ZFC')
    subplot(3,1,1);
    title('dB playback for MDL and MML version of ZFC (18sec)', 'fontsize', 18)
    xlim([0 endtime])
```

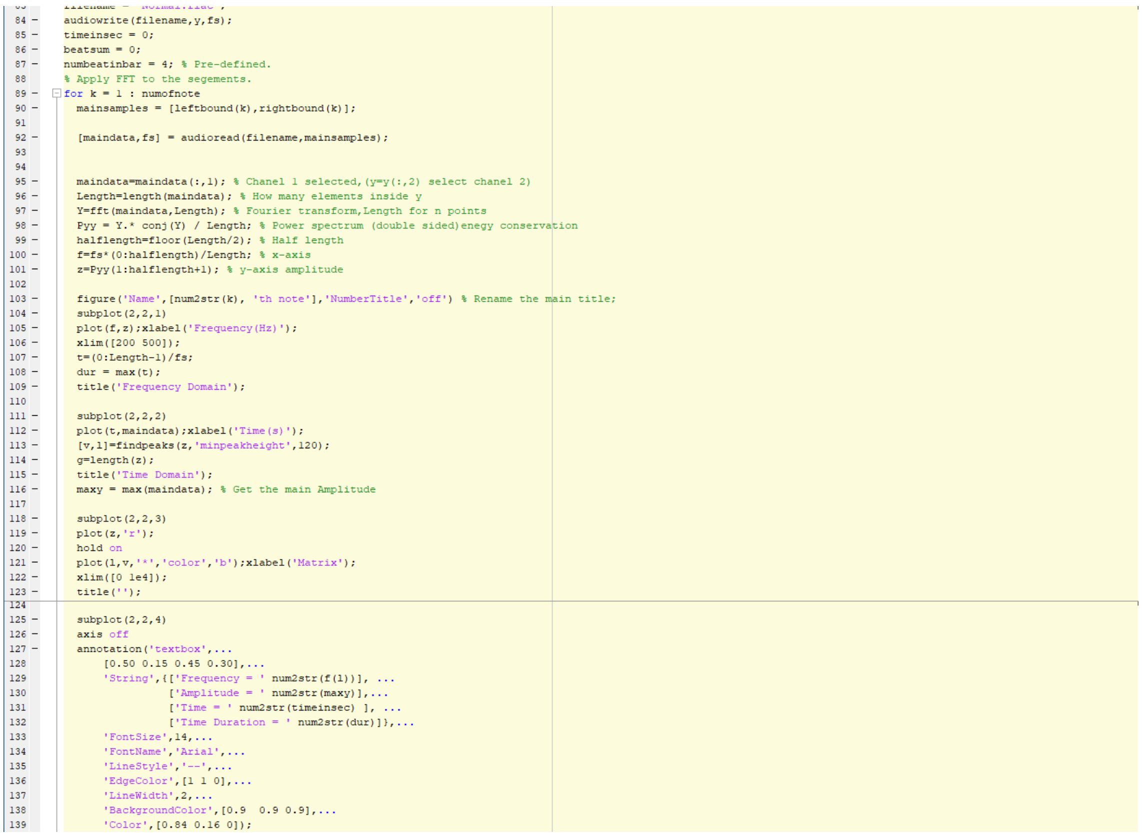
### Algorithm 2, 9

Extra lines for Algorithm 9, related functions were from http://kenschutte.com/midi:



```matlab
% Final Version
% Initialization~
close all; clear all; clc; % Close Everything.
bits=16; % Initialize the bits.
thresg = 0;
leftend = 0; % Initialize the leftend
rightend = 0; % Initialize the rightend

midi = readmidi('TwinkleTwinkleLittleStarNormal.mid');
[y, fs] = midi2audio(midi);
y = midi2audio(midi, fs, 'sine');
y = .95.*y./max(abs(y));
audiowrite('sample.wav', y, fs);

% Import File~
[y,fs]=audioread('sample.wav'); % Audio Read the file.
info = audioinfo('sample.wav'); % Get the audio information. (% disp(info); % Display full info)
duration = info.Duration; % Get duration of the music track.
fs=info.SampleRate; % Get Sampling rate.
lengthofy = length(y); % Get the length of file.
t=linspace(0,lengthofy/fs,lengthofy); % linear spacing the time using the sampling rate.
sound(y,fs) % Play the music track

% Overal Plotting~
figure('Name',info.Filename,'NumberTitle','off') % Rename the main title
subplot(2,1,1)
plot(t,y) % Time Domain Plot
title('Time Domain Plot')
```

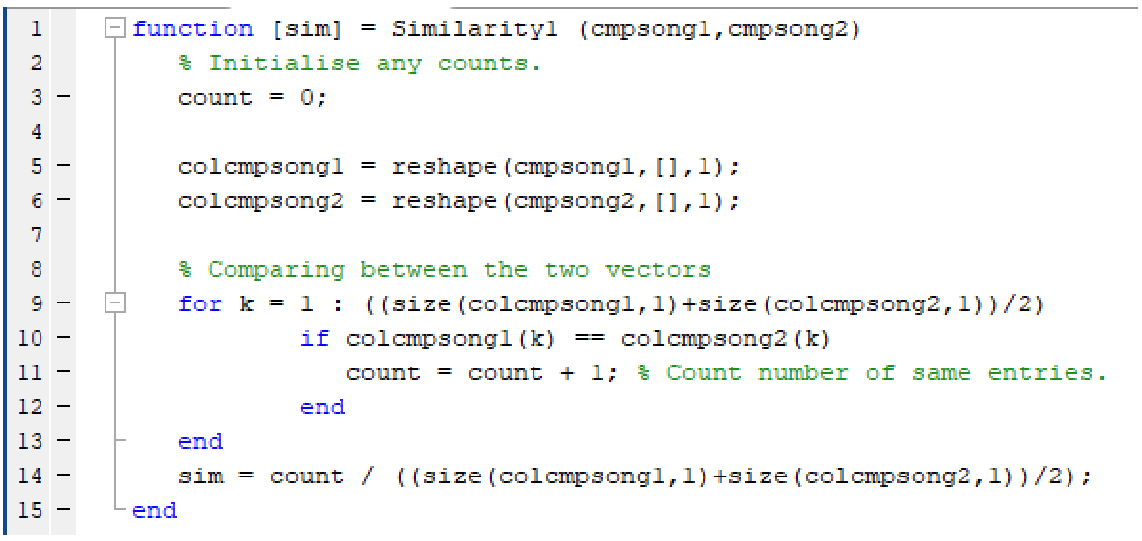
FFT:



```
filename = 'Normal.flac';
audiowrite(filename,y,fs);
timeinsec = 0;
beatsum = 0;
numbeatinbar = 4; % Pre-defined.
% Apply FFT to the segements.
for k = 1 : numofnote
  mainsamples = [leftbound(k),rightbound(k)];

  [maindata,fs] = audioread(filename,mainsamples);


  maindata=maindata(:,1); % Chanel 1 selected,(y=y(:,2) select chanel 2)
  Length=length(maindata); % How many elements inside y
  Y=fft(maindata,Length); % Fourier transform,Length for n points
  Pyy = Y.* conj(Y) / Length; % Power spectrum (double sided)enegy conservation
  halflength=floor(Length/2); % Half length
  f=fs*(0:halflength)/Length; % x-axis
  z=Pyy(1:halflength+1); % y-axis amplitude

  figure('Name',[num2str(k), 'th note'],'NumberTitle','off') % Rename the main title;
  subplot(2,2,1)
  plot(f,z);xlabel('Frequency(Hz)');
  xlim([200 500]);
  t=(0:Length-1)/fs;
  dur = max(t);
  title('Frequency Domain');

  subplot(2,2,2)
  plot(t,maindata);xlabel('Time(s)');
  [v,l]=findpeaks(z,'minpeakheight',120);
  g=length(z);
  title('Time Domain');
  maxy = max(maindata); % Get the main Amplitude

  subplot(2,2,3)
  plot(z,'r');
  hold on
  plot(l,v,'*','color','b');xlabel('Matrix');
  xlim([0 1e4]);
  title('');

  subplot(2,2,4)
  axis off
  annotation('textbox',...
      [0.50 0.15 0.45 0.30],...
      'String',{['Frequency = ' num2str(f(1))], ...
                ['Amplitude = ' num2str(maxy)],...
                ['Time = ' num2str(timeinsec) ], ...
                ['Time Duration = ' num2str(dur)]},...
      'FontSize',14,...
      'FontName','Arial',...
      'LineStyle','--',...
      'EdgeColor',[1 1 0],...
      'LineWidth',2,...
      'BackgroundColor',[0.9  0.9 0.9],...
      'Color',[0.84 0.16 0]);
```

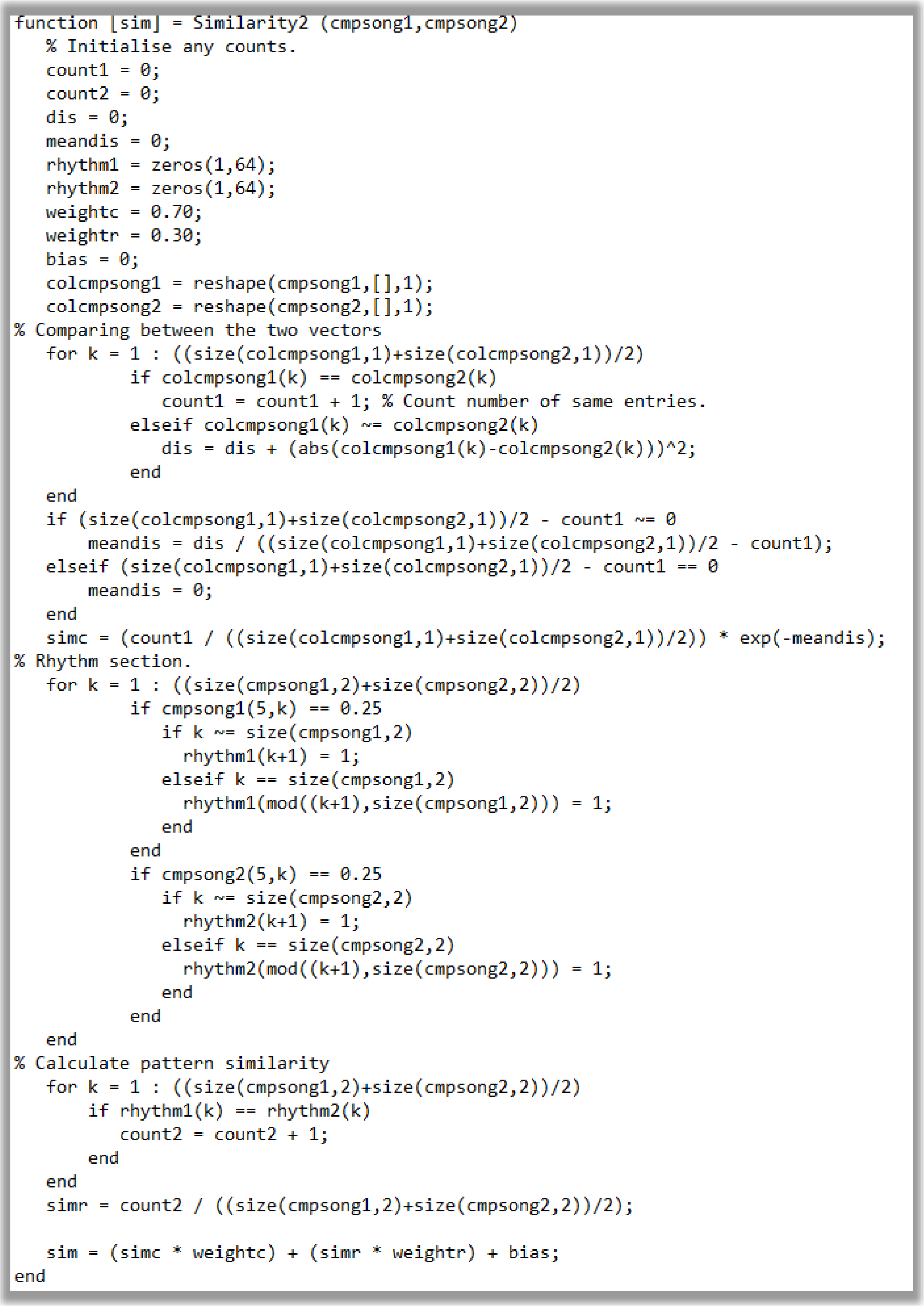
## Algorithm 3-4,7-8

SC1:



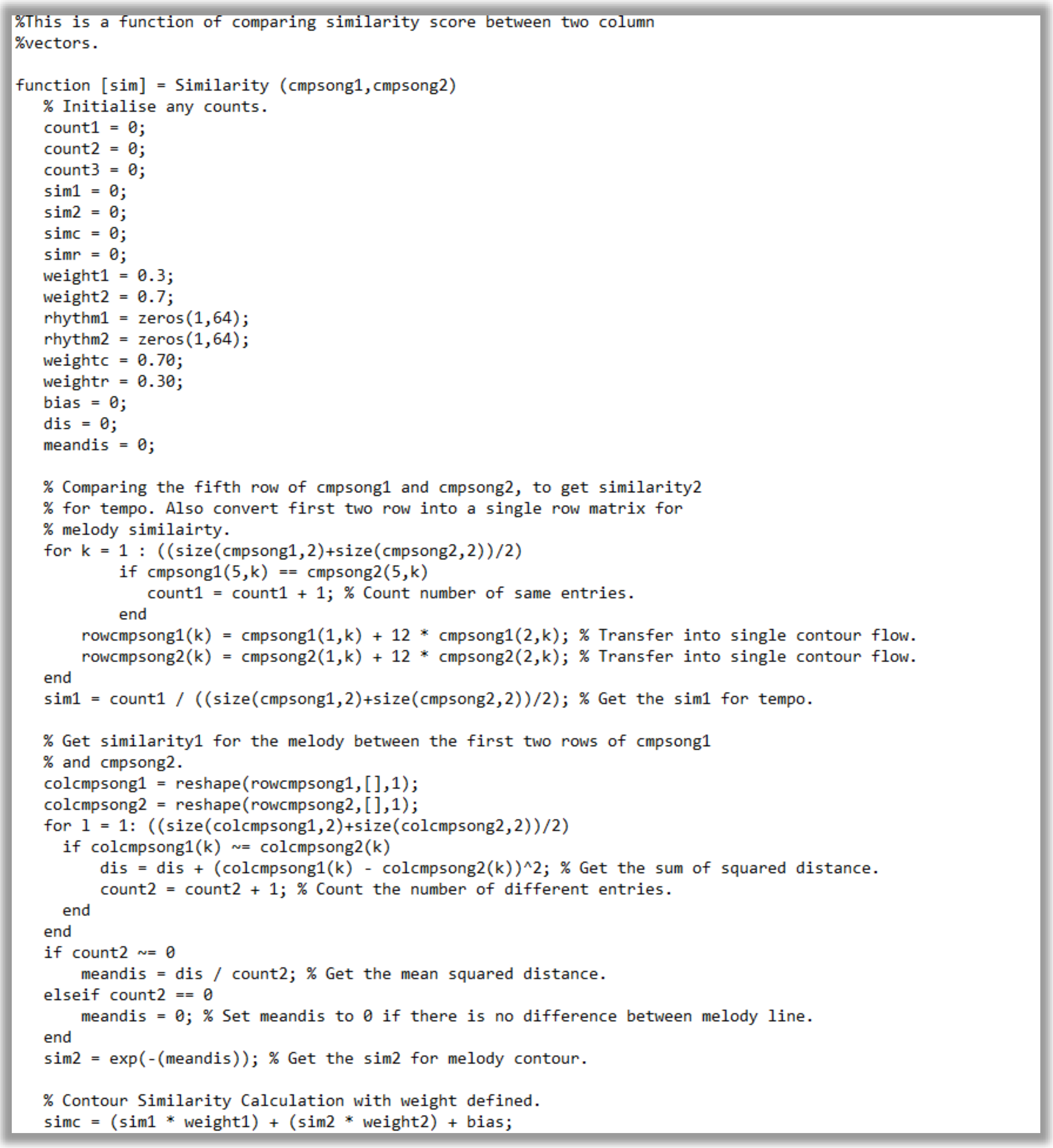
```
function [sim] = Similarity1 (cmpsong1,cmpsong2)
    % Initialise any counts.
    count = 0;

    colcmpsong1 = reshape(cmpsong1,[],1);
    colcmpsong2 = reshape(cmpsong2,[],1);

    % Comparing between the two vectors
    for k = 1 : ((size(colcmpsong1,1)+size(colcmpsong2,1))/2)
            if colcmpsong1(k) == colcmpsong2(k)
                count = count + 1; % Count number of same entries.
            end
    end
    sim = count / ((size(colcmpsong1,1)+size(colcmpsong2,1))/2);
end
```

SC2:



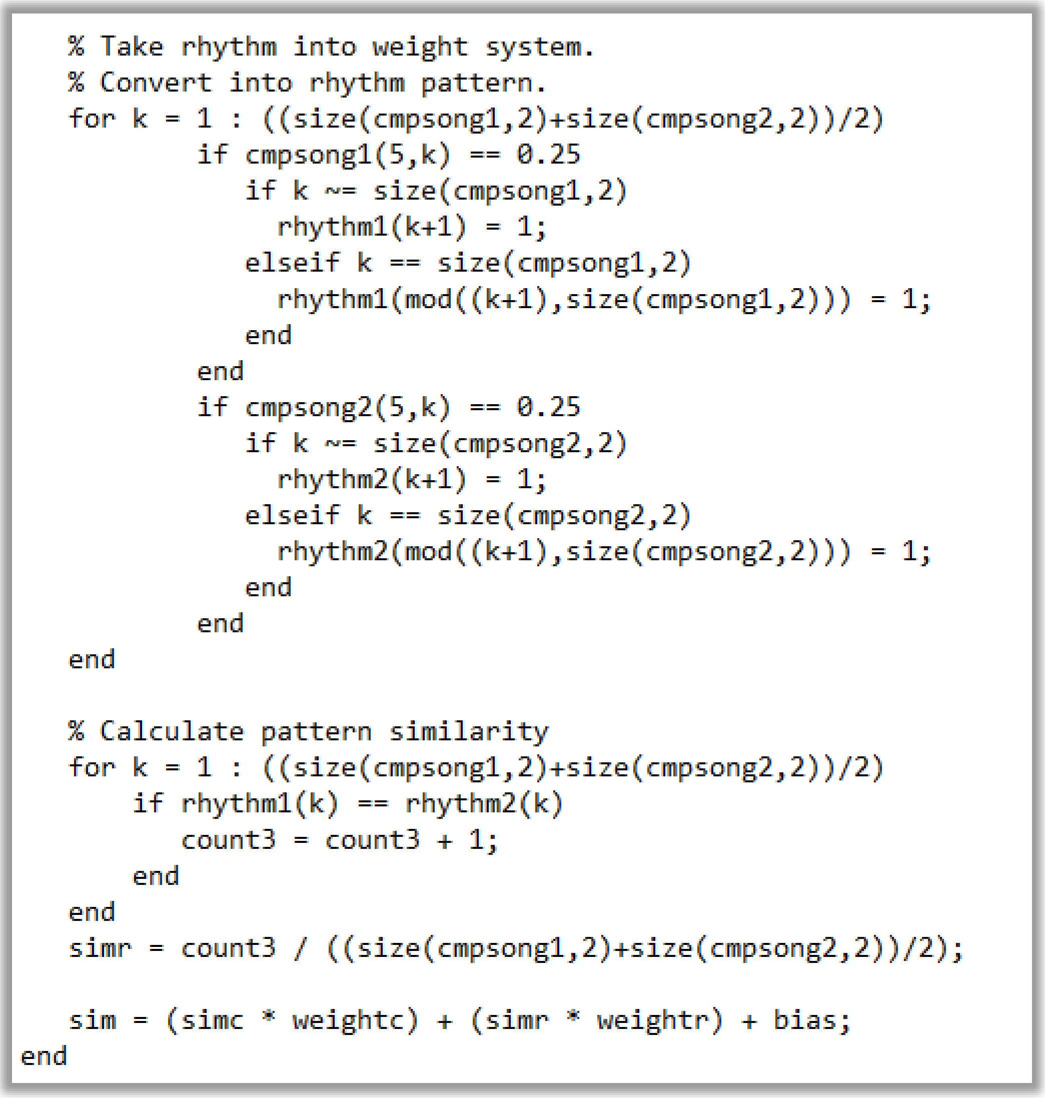
```
function [sim] = Similarity2 (cmpsong1,cmpsong2)
   % Initialise any counts.
   count1 = 0;
   count2 = 0;
   dis = 0;
   meandis = 0;
   rhythm1 = zeros(1,64);
   rhythm2 = zeros(1,64);
   weightc = 0.70;
   weightr = 0.30;
   bias = 0;
   colcmpsong1 = reshape(cmpsong1,[],1);
   colcmpsong2 = reshape(cmpsong2,[],1);
% Comparing between the two vectors
   for k = 1 : ((size(colcmpsong1,1)+size(colcmpsong2,1))/2)
           if colcmpsong1(k) == colcmpsong2(k)
              count1 = count1 + 1; % Count number of same entries.
           elseif colcmpsong1(k) ~= colcmpsong2(k)
              dis = dis + (abs(colcmpsong1(k)-colcmpsong2(k)))^2;
           end
   end
   if (size(colcmpsong1,1)+size(colcmpsong2,1))/2 - count1 ~= 0
       meandis = dis / ((size(colcmpsong1,1)+size(colcmpsong2,1))/2 - count1);
   elseif (size(colcmpsong1,1)+size(colcmpsong2,1))/2 - count1 == 0
       meandis = 0;
   end
   simc = (count1 / ((size(colcmpsong1,1)+size(colcmpsong2,1))/2)) * exp(-meandis);
% Rhythm section.
   for k = 1 : ((size(cmpsong1,2)+size(cmpsong2,2))/2)
           if cmpsong1(5,k) == 0.25
              if k ~= size(cmpsong1,2)
                rhythm1(k+1) = 1;
              elseif k == size(cmpsong1,2)
                rhythm1(mod((k+1),size(cmpsong1,2))) = 1;
              end
           end
           if cmpsong2(5,k) == 0.25
              if k ~= size(cmpsong2,2)
                rhythm2(k+1) = 1;
              elseif k == size(cmpsong2,2)
                rhythm2(mod((k+1),size(cmpsong2,2))) = 1;
              end
           end
   end
% Calculate pattern similarity
   for k = 1 : ((size(cmpsong1,2)+size(cmpsong2,2))/2)
       if rhythm1(k) == rhythm2(k)
          count2 = count2 + 1;
       end
   end
   simr = count2 / ((size(cmpsong1,2)+size(cmpsong2,2))/2);

   sim = (simc * weightc) + (simr * weightr) + bias;
end
```

SC3:



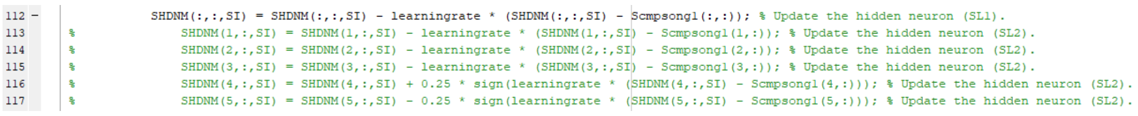
```
%This is a function of comparing similarity score between two column
%vectors.

function [sim] = Similarity (cmpsong1,cmpsong2)
   % Initialise any counts.
   count1 = 0;
   count2 = 0;
   count3 = 0;
   sim1 = 0;
   sim2 = 0;
   simc = 0;
   simr = 0;
   weight1 = 0.3;
   weight2 = 0.7;
   rhythm1 = zeros(1,64);
   rhythm2 = zeros(1,64);
   weightc = 0.70;
   weightr = 0.30;
   bias = 0;
   dis = 0;
   meandis = 0;

   % Comparing the fifth row of cmpsong1 and cmpsong2, to get similarity2
   % for tempo. Also convert first two row into a single row matrix for
   % melody similairty.
   for k = 1 : ((size(cmpsong1,2)+size(cmpsong2,2))/2)
           if cmpsong1(5,k) == cmpsong2(5,k)
              count1 = count1 + 1; % Count number of same entries.
           end
       rowcmpsong1(k) = cmpsong1(1,k) + 12 * cmpsong1(2,k); % Transfer into single contour flow.
       rowcmpsong2(k) = cmpsong2(1,k) + 12 * cmpsong2(2,k); % Transfer into single contour flow.
   end
   sim1 = count1 / ((size(cmpsong1,2)+size(cmpsong2,2))/2); % Get the sim1 for tempo.

   % Get similarity1 for the melody between the first two rows of cmpsong1
   % and cmpsong2.
   colcmpsong1 = reshape(rowcmpsong1,[],1);
   colcmpsong2 = reshape(rowcmpsong2,[],1);
   for l = 1: ((size(colcmpsong1,2)+size(colcmpsong2,2))/2)
     if colcmpsong1(k) ~= colcmpsong2(k)
         dis = dis + (colcmpsong1(k) - colcmpsong2(k))^2; % Get the sum of squared distance.
         count2 = count2 + 1; % Count the number of different entries.
     end
   end
   if count2 ~= 0
       meandis = dis / count2; % Get the mean squared distance.
   elseif count2 == 0
       meandis = 0; % Set meandis to 0 if there is no difference between melody line.
   end
   sim2 = exp(-(meandis)); % Get the sim2 for melody contour.

   % Contour Similarity Calculation with weight defined.
   simc = (sim1 * weight1) + (sim2 * weight2) + bias;
```

RC&SR:



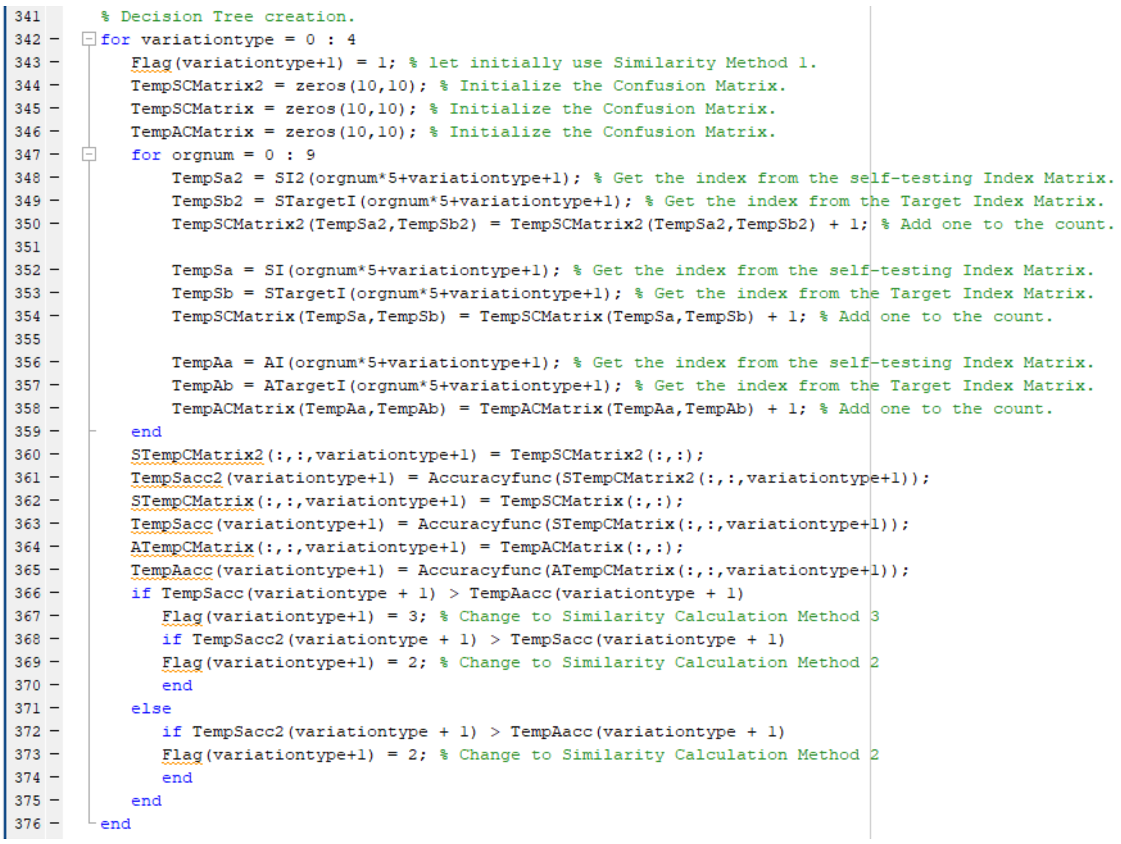
```
   % Take rhythm into weight system.
   % Convert into rhythm pattern.
   for k = 1 : ((size(cmpsong1,2)+size(cmpsong2,2))/2)
           if cmpsong1(5,k) == 0.25
              if k ~= size(cmpsong1,2)
                rhythm1(k+1) = 1;
              elseif k == size(cmpsong1,2)
                rhythm1(mod((k+1),size(cmpsong1,2))) = 1;
              end
           end
           if cmpsong2(5,k) == 0.25
              if k ~= size(cmpsong2,2)
                rhythm2(k+1) = 1;
              elseif k == size(cmpsong2,2)
                rhythm2(mod((k+1),size(cmpsong2,2))) = 1;
              end
           end
   end

   % Calculate pattern similarity
   for k = 1 : ((size(cmpsong1,2)+size(cmpsong2,2))/2)
       if rhythm1(k) == rhythm2(k)
          count3 = count3 + 1;
       end
   end
   simr = count3 / ((size(cmpsong1,2)+size(cmpsong2,2))/2);

   sim = (simc * weightc) + (simr * weightr) + bias;
end
```

## SL1&SL2:



```
            SHDNM(:,:,SI) = SHDNM(:,:,SI) - learningrate * (SHDNM(:,:,SI) - Scmpsong1(:,:)); % Update the hidden neuron (SL1).
%               SHDNM(1,:,SI) = SHDNM(1,:,SI) - learningrate * (SHDNM(1,:,SI) - Scmpsong1(1,:)); % Update the hidden neuron (SL2).
%               SHDNM(2,:,SI) = SHDNM(2,:,SI) - learningrate * (SHDNM(2,:,SI) - Scmpsong1(2,:)); % Update the hidden neuron (SL2).
%               SHDNM(3,:,SI) = SHDNM(3,:,SI) - learningrate * (SHDNM(3,:,SI) - Scmpsong1(3,:)); % Update the hidden neuron (SL2).
%               SHDNM(4,:,SI) = SHDNM(4,:,SI) + 0.25 * sign(learningrate * (SHDNM(4,:,SI) - Scmpsong1(4,:))); % Update the hidden neuron (SL2).
%               SHDNM(5,:,SI) = SHDNM(5,:,SI) - 0.25 * sign(learningrate * (SHDNM(5,:,SI) - Scmpsong1(5,:))); % Update the hidden neuron (SL2).
```

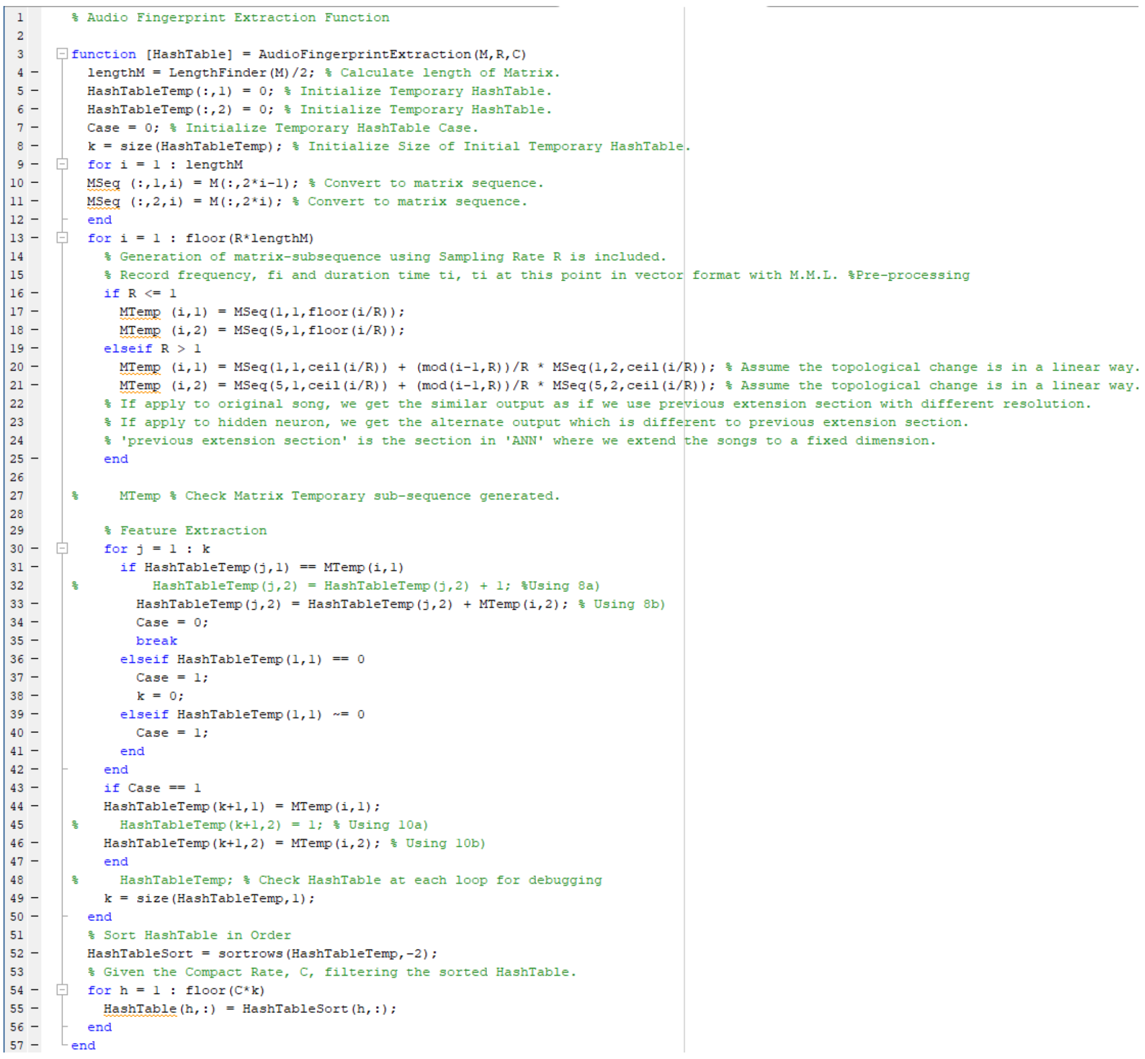
## Strategy Tree Generation for RL-EML:



```
% Decision Tree creation.
for variationtype = 0 : 4
   Flag(variationtype+1) = 1; % let initially use Similarity Method 1.
   TempSCMatrix2 = zeros(10,10); % Initialize the Confusion Matrix.
   TempSCMatrix = zeros(10,10); % Initialize the Confusion Matrix.
   TempACMatrix = zeros(10,10); % Initialize the Confusion Matrix.
   for orgnum = 0 : 9
       TempSa2 = SI2(orgnum*5+variationtype+1); % Get the index from the self-testing Index Matrix.
       TempSb2 = STargetI(orgnum*5+variationtype+1); % Get the index from the Target Index Matrix.
       TempSCMatrix2(TempSa2,TempSb2) = TempSCMatrix2(TempSa2,TempSb2) + 1; % Add one to the count.

       TempSa = SI(orgnum*5+variationtype+1); % Get the index from the self-testing Index Matrix.
       TempSb = STargetI(orgnum*5+variationtype+1); % Get the index from the Target Index Matrix.
       TempSCMatrix(TempSa,TempSb) = TempSCMatrix(TempSa,TempSb) + 1; % Add one to the count.

       TempAa = AI(orgnum*5+variationtype+1); % Get the index from the self-testing Index Matrix.
       TempAb = ATargetI(orgnum*5+variationtype+1); % Get the index from the Target Index Matrix.
       TempACMatrix(TempAa,TempAb) = TempACMatrix(TempAa,TempAb) + 1; % Add one to the count.
   end
   STempCMatrix2(:,:,variationtype+1) = TempSCMatrix2(:,:);
   TempSacc2(variationtype+1) = Accuracyfunc(STempCMatrix2(:,:,variationtype+1));
   STempCMatrix(:,:,variationtype+1) = TempSCMatrix(:,:);
   TempSacc(variationtype+1) = Accuracyfunc(STempCMatrix(:,:,variationtype+1));
   ATempCMatrix(:,:,variationtype+1) = TempACMatrix(:,:);
   TempAacc(variationtype+1) = Accuracyfunc(ATempCMatrix(:,:,variationtype+1));
   if TempSacc(variationtype + 1) > TempAacc(variationtype + 1)
      Flag(variationtype+1) = 3; % Change to Similarity Calculation Method 3
      if TempSacc2(variationtype + 1) > TempSacc(variationtype + 1)
      Flag(variationtype+1) = 2; % Change to Similarity Calculation Method 2
      end
   else
      if TempSacc2(variationtype + 1) > TempAacc(variationtype + 1)
      Flag(variationtype+1) = 2; % Change to Similarity Calculation Method 2
      end
   end
end
```

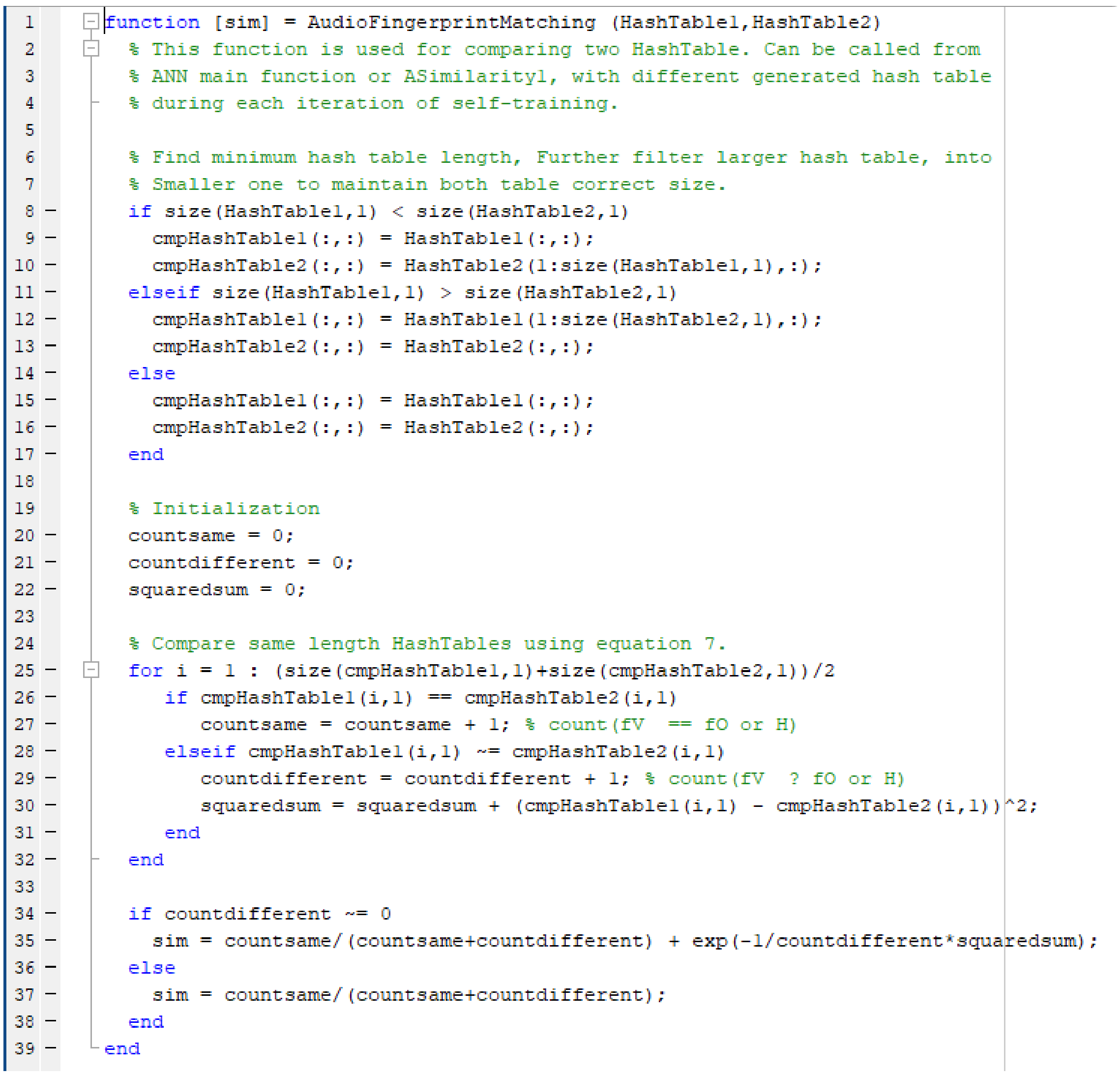
## Algorithm 5-6

AF-Extraction:



```matlab
% Audio Fingerprint Extraction Function

function [HashTable] = AudioFingerprintExtraction(M,R,C)
  lengthM = LengthFinder(M)/2; % Calculate length of Matrix.
  HashTableTemp(:,1) = 0; % Initialize Temporary HashTable.
  HashTableTemp(:,2) = 0; % Initialize Temporary HashTable.
  Case = 0; % Initialize Temporary HashTable Case.
  k = size(HashTableTemp); % Initialize Size of Initial Temporary HashTable.
  for i = 1 : lengthM
  MSeq (:,1,i) = M(:,2*i-1); % Convert to matrix sequence.
  MSeq (:,2,i) = M(:,2*i); % Convert to matrix sequence.
  end
  for i = 1 : floor(R*lengthM)
    % Generation of matrix-subsequence using Sampling Rate R is included.
    % Record frequency, fi and duration time ti, ti at this point in vector format with M.M.L. %Pre-processing
    if R <= 1
      MTemp (i,1) = MSeq(1,1,floor(i/R));
      MTemp (i,2) = MSeq(5,1,floor(i/R));
    elseif R > 1
      MTemp (i,1) = MSeq(1,1,ceil(i/R)) + (mod(i-1,R))/R * MSeq(1,2,ceil(i/R)); % Assume the topological change is in a linear way.
      MTemp (i,2) = MSeq(5,1,ceil(i/R)) + (mod(i-1,R))/R * MSeq(5,2,ceil(i/R)); % Assume the topological change is in a linear way.
    % If apply to original song, we get the similar output as if we use previous extension section with different resolution.
    % If apply to hidden neuron, we get the alternate output which is different to previous extension section.
    % 'previous extension section' is the section in 'ANN' where we extend the songs to a fixed dimension.
    end

%     MTemp % Check Matrix Temporary sub-sequence generated.

    % Feature Extraction
    for j = 1 : k
      if HashTableTemp(j,1) == MTemp(i,1)
%         HashTableTemp(j,2) = HashTableTemp(j,2) + 1; %Using 8a)
        HashTableTemp(j,2) = HashTableTemp(j,2) + MTemp(i,2); % Using 8b)
        Case = 0;
        break
      elseif HashTableTemp(1,1) == 0
        Case = 1;
        k = 0;
      elseif HashTableTemp(1,1) ~= 0
        Case = 1;
      end
    end
    if Case == 1
    HashTableTemp(k+1,1) = MTemp(i,1);
%     HashTableTemp(k+1,2) = 1; % Using 10a)
    HashTableTemp(k+1,2) = MTemp(i,2); % Using 10b)
    end
%     HashTableTemp; % Check HashTable at each loop for debugging
    k = size(HashTableTemp,1);
  end
  % Sort HashTable in Order
  HashTableSort = sortrows(HashTableTemp,-2);
  % Given the Compact Rate, C, filtering the sorted HashTable.
  for h = 1 : floor(C*k)
    HashTable(h,:) = HashTableSort(h,:);
  end
end
```

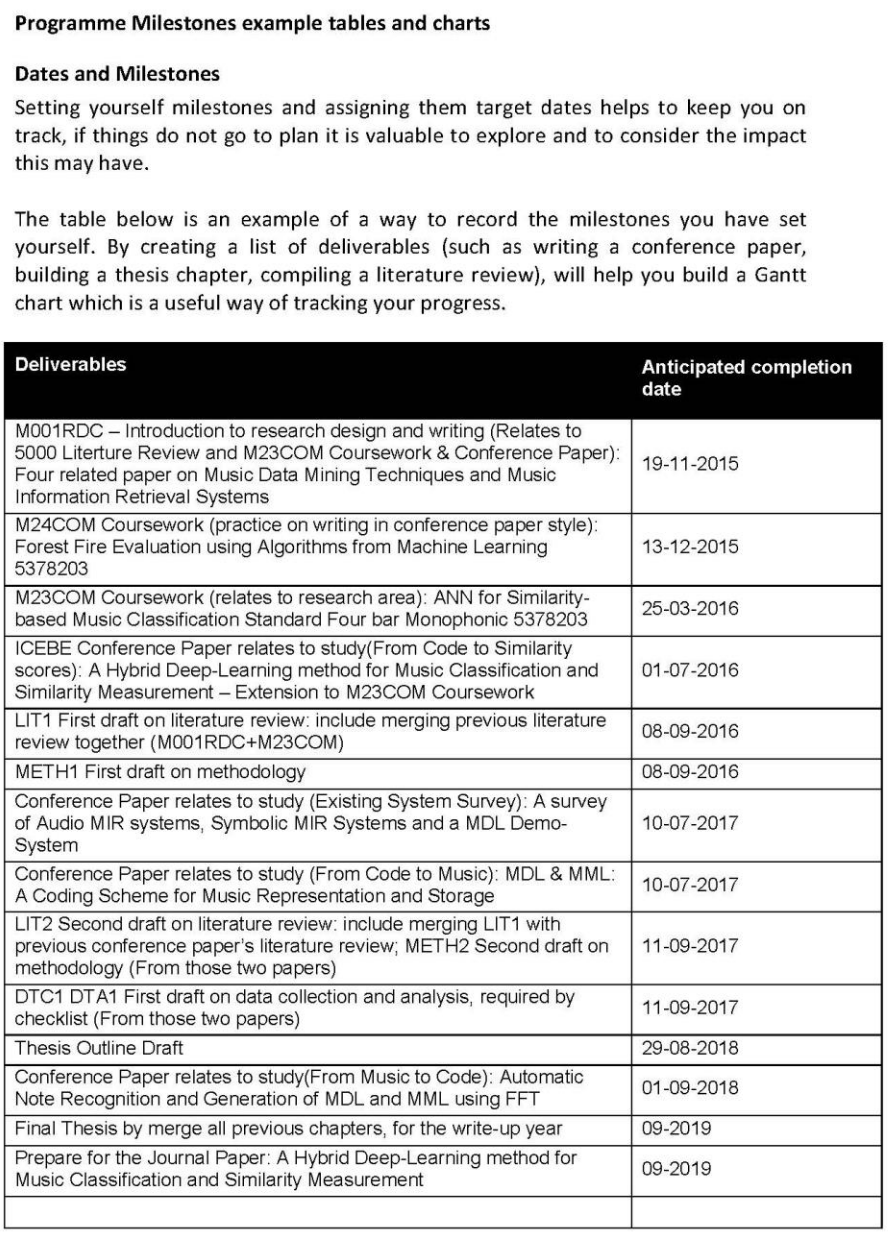
AF-Matching:



```matlab
function [sim] = AudioFingerprintMatching (HashTable1,HashTable2)
  % This function is used for comparing two HashTable. Can be called from
  % ANN main function or ASimilarity1, with different generated hash table
  % during each iteration of self-training.

  % Find minimum hash table length, Further filter larger hash table, into
  % Smaller one to maintain both table correct size.
  if size(HashTable1,1) < size(HashTable2,1)
    cmpHashTable1(:,:) = HashTable1(:,:);
    cmpHashTable2(:,:) = HashTable2(1:size(HashTable1,1),:);
  elseif size(HashTable1,1) > size(HashTable2,1)
    cmpHashTable1(:,:) = HashTable1(1:size(HashTable2,1),:);
    cmpHashTable2(:,:) = HashTable2(:,:);
  else
    cmpHashTable1(:,:) = HashTable1(:,:);
    cmpHashTable2(:,:) = HashTable2(:,:);
  end

  % Initialization
  countsame = 0;
  countdifferent = 0;
  squaredsum = 0;

  % Compare same length HashTables using equation 7.
  for i = 1 : (size(cmpHashTable1,1)+size(cmpHashTable2,1))/2
     if cmpHashTable1(i,1) == cmpHashTable2(i,1)
        countsame = countsame + 1; % count(fV  == fO or H)
     elseif cmpHashTable1(i,1) ~= cmpHashTable2(i,1)
        countdifferent = countdifferent + 1; % count(fV  ? fO or H)
        squaredsum = squaredsum + (cmpHashTable1(i,1) - cmpHashTable2(i,1))^2;
     end
  end

  if countdifferent ~= 0
    sim = countsame/(countsame+countdifferent) + exp(-1/countdifferent*squaredsum);
  else
    sim = countsame/(countsame+countdifferent);
  end
end
```

## Appendix C. Project Management & Others

**Project Milestone & Gantt Chart for Time Management:**



**Programme Milestones example tables and charts**

**Dates and Milestones**

Setting yourself milestones and assigning them target dates helps to keep you on track, if things do not go to plan it is valuable to explore and to consider the impact this may have.

The table below is an example of a way to record the milestones you have set yourself. By creating a list of deliverables (such as writing a conference paper, building a thesis chapter, compiling a literature review), will help you build a Gantt chart which is a useful way of tracking your progress.

| Deliverables | Anticipated completion date |
|---|---|
| M001RDC – Introduction to research design and writing (Relates to 5000 Literture Review and M23COM Coursework & Conference Paper): Four related paper on Music Data Mining Techniques and Music Information Retrieval Systems | 19-11-2015 |
| M24COM Coursework (practice on writing in conference paper style): Forest Fire Evaluation using Algorithms from Machine Learning 5378203 | 13-12-2015 |
| M23COM Coursework (relates to research area): ANN for Similarity-based Music Classification Standard Four bar Monophonic 5378203 | 25-03-2016 |
| ICEBE Conference Paper relates to study(From Code to Similarity scores): A Hybrid Deep-Learning method for Music Classification and Similarity Measurement – Extension to M23COM Coursework | 01-07-2016 |
| LIT1 First draft on literature review: include merging previous literature review together (M001RDC+M23COM) | 08-09-2016 |
| METH1 First draft on methodology | 08-09-2016 |
| Conference Paper relates to study (Existing System Survey): A survey of Audio MIR systems, Symbolic MIR Systems and a MDL Demo-System | 10-07-2017 |
| Conference Paper relates to study (From Code to Music): MDL & MML: A Coding Scheme for Music Representation and Storage | 10-07-2017 |
| LIT2 Second draft on literature review: include merging LIT1 with previous conference paper's literature review; METH2 Second draft on methodology (From those two papers) | 11-09-2017 |
| DTC1 DTA1 First draft on data collection and analysis, required by checklist (From those two papers) | 11-09-2017 |
| Thesis Outline Draft | 29-08-2018 |
| Conference Paper relates to study(From Music to Code): Automatic Note Recognition and Generation of MDL and MML using FFT | 01-09-2018 |
| Final Thesis by merge all previous chapters, for the write-up year | 09-2019 |
| Prepare for the Journal Paper: A Hybrid Deep-Learning method for Music Classification and Similarity Measurement | 09-2019 |
| | |

**Programme Gantt Charts**

*Phase 1*

| Year:2015-2016 | Months | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Task** | **9** | **10** | **11** | **12** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| M001RDC – Introduction to research design and writing (Relates to 5000 Literture Review and M23COM Coursework & Conference Paper): Four related paper on Music Data Mining Techniques and Music Information Retrieval Systems | 🟥 | 🟥 | 🟥 | | | | | | | | | |
| M24COM Coursework (practice on writing in conference paper style): Forest Fire Evaluation using Algorithms from Machine Learning 5378203 | | | 🟥 | 🟥 | | | | | | | | |
| M23COM Coursework (relates to research area): ANN for Similarity-based Music Classification Standard Four bar Monophonic 5378203 | | | | | | 🟥 | 🟥 | | | | | |
| ICEBE Conference Paper relates to study(From Code to Similarity scores): A Hybrid Deep-Learning method for Music Classification and Similarity Measurement – Extension to M23COM Coursework | | | | | | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | | |
| LIT1 First draft on literature review: include merging previous literature review together (M001RDC+M23COM) | 🟥 | 🟥 | 🟥 | | 🟥 | 🟥 | | | | | 🟥 | 🟥 |
| METH1 First draft on methodology | | | | 🟥 | 🟥 | | | | | | 🟥 | 🟥 |

*Phase 2*

| Year: 2016-2017 | Months | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Task** | **9** | **10** | **11** | **12** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Conference Paper relates to study(Existing System Survey): Title to be confirmed | | 🟥 | | | 🟥 | 🟥 | | | 🟥 | | 🟥 | |
| Conference Paper relates to study(From Code to Music): Title to be confirmed | | 🟥 | | | | | 🟥 | 🟥 | | 🟥 | 🟥 | |
| LIT2 Second draft on literature review: include merging LIT1 with previous conference paper's literature review (From those two papers) | 🟥 | | 🟥 | 🟥 | | | | | | | | 🟥 |
| METH2 Second draft on methodology (From those two papers) | 🟥 | | 🟥 | 🟥 | | | | | | | | 🟥 |
| DTC1 DTA1 First draft on data collection and analysis, required by checklist (From those two papers) | | | 🟥 | 🟥 | 🟥 | | 🟥 | | | | | 🟥 |
| Prepare for the Journal Paper: A Hybrid Deep-Learning method for Music Classification and Similarity Measurement | | | | | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |

*Phase 3*

| **Year: 2017-2018** | **Months** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Task** | **9** | **10** | **11** | **12** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Thesis Outline Draft | | | | 🟥 | 🟥 | | | | | | 🟥 | 🟥 |
| Conference Paper relates to study(From Music to Code): Automatic Note Recognition and Generation of MDL and MML using FFT | | | | | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | | |
| Final Thesis by merge all previous chapters, for the write-up year | | | | | | | | | | | | 🟥 |
| Prepare for the Journal Paper: A Hybrid Deep-Learning method for Music Classification and Similarity Measurement | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |

## Risk Assessment Form:

1

Content removed on data protection grounds

Content removed on data protection grounds

Content removed on data protection grounds