UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

Engineering School

Department of Computing, Electronics, and Mechatronics

DOCTORATE IN INTELLIGENT SYSTEMS

Creating a creator: a methodology for music data analysis, feature visualization, and automatic music composition

Thesis submitted as partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Omar López Rincón

Advisor:

Oleg Starostenko Basarab PhD.

Santa Catarina Mártir, San Andrés Cholula, Puebla

Table of Contents

List of Figures

Abstract

Music algorithmic composition has recently become an area of prestigious research projects such as Google's Magenta project, Sony´s CSL Lab in Paris, and Aiva at Luxembourg, which are pushing the boundaries in automatic music generation. Computer music generation is the process of composing music through an algorithmic approach or the use of artificial intelligence. In the filmmaking and gaming industries, *startups* or starting companies can compete with the bigger ones relying on generative methods to help them increase productivity and rely solely on their creativity. Nowadays, the generative methods for music pieces creation had been tested with machine learning methods like Recurrent Neural Networks and Markovian approaches with different outcomes. A specific problem with Recurrent Neural Networks is the amount of time it takes to train an architecture aiming at a particular subset of data. Another problem with the supervised learning approaches is the absence of accessible datasets with proper labeling of the music features. Additionally, there are no metrics or enough standard samples of the different aspects of music talking about melody, harmony, and rhythm as well as there are not metrics that can be used to quantify music perception.

In this research, we introduce a simple and efficient methodology for analyzing and testing harmonic features of music using different quantitative metrics. Notably, the proposed approach normalizes data in MIDI files by 12-dimensional vector descriptors extracted from tonality as well as it is used for dimensionality reduction and visualization of extracted music data by 3D feature vector projections. This projection is achieved through a non-overlapping sliding window through the composition, harmonic features are found in the music piece, and three-dimension projection creates a quantitative profile of a composition, which correlates the tone similarities along with the music piece.

Another contribution consists of designing a methodology to extract the rhythm and harmony of musical pieces and provide their recombination. These extractions are done through an algorithmic compression approach to finding the descriptors that can encode the development of the music piece. Due to the music sequences do not have the same length as well as the size of the songs are different, the process of coupling the harmony descriptors to

the rhythm extracted from the MIDI file is proposed. This coupling is done using a genetic algorithm to take advantage of artificial intelligence in automatic music generation.

Finally, we propose a novel technique for generating new music composition by replacing the existing harmony descriptors of the MIDI file with a new harmony generated by the genetic algorithm and combining it with a rhythm of other compositions providing in this way adjustment of the new music piece to a particular genre. All the proposed approaches have been tested, evaluated, and compared with existed prototypes ensuring their high quality and efficiency during feature extraction, analysis, visualization, and automatic generation of polyphonic music compositions.

Resumen

La generación algorítmica de música recientemente se ha convertido en área de proyectos de investigación prestigiosos como Magenta de Google, el laboratorio en Paris de Sony CSL y Aiva en Luxemburgo, el cual empuja las barreras en generación automática de música. La generación musical por computadora es el proceso de componer música a través de un enfoque algorítmico o con el uso de inteligencia artificial. En las industrias de cine y videojuegos se encuentran las llamadas startups o compañías emergentes podrían competir con las más grandes basándose en métodos generativos para ayudarles a incrementar la productividad y depender únicamente de la creatividad. Hoy en día, los métodos generativos de piezas musicales han sido probados con métodos de aprendizaje de máquina como redes recurrentes o métodos Markovianos con diferentes resultados. Un problema específico con las redes recurrentes es el tiempo que toma entrenar el modelo con un subconjunto de datos. Otro problema con métodos supervisados es la ausencia de datos suficientes etiquetados con las características musicales. Adicionalmente, no hay métricas o ejemplos suficientes de las diferentes características de música como melodía, armonía y ritmo, así como no hay métricas que reflejen la percepción musical.

En esta investigación, presentamos una metodología simple y eficiente para analizar y probar características armónicas de música utilizando diferentes métricas cualitativas. Notablemente, el enfoque propuesto normaliza los datos que se encuentran en archivos MIDI en vectores descriptores de 12 dimensiones extraídos de la tonalidad y utilizados para reducción de dimensionalidad y visualización de los datos musicales extraídos proyectados en vectores de tres dimensiones. Esta proyección se logra mediante ventanas deslizantes no

traslapadas a través de la composición, las características armónicas se encuentran en la pieza musical y la proyección tridimensional crea un perfil cuantitativo de la composición, el cual esta correlacionado con similitudes de tono en la pieza.

Otra contribución consiste en el diseño de una metodología para extraer el ritmo y armonía de una pieza musical y realizar una recombinación de estas características. Las extracciones se llevan a cabo a través de una compresión algorítmica para encontrar los descriptores que pueden codificar el desarrollo de la pieza musical. Como las piezas musicales no poseen la misma longitud y así como las canciones son de duraciones diferentes, se propone el proceso de acoplamiento de los descriptores armónicas al ritmo extraído de los archivos MIDI. Este acoplamiento se logra utilizando un algoritmo genético para aprovechar las ventajas de la inteligencia artificial in la generación automática de música.

Finalmente, se propone una técnica novedosa de generación de composiciones musicales por reemplazo de los descriptores armónicos de un archivo MIDI con una armonía generada por el algoritmo genético y combinándola con el ritmo de otra composición haciendo un ajuste para generar una nueva pieza musical. Todas las propuestas fueron probadas, evaluadas y comparadas con prototipos existentes asegurando la alta calidad y eficiencia durante la extracción de características, análisis, visualización y generación automática de composiciones musicales polifónicos y poli-instrumentales.

Chapter 1 Introduction

1.1 Problem definition and Motivation

Automatic music generation is not a new area of research. The first attempts can be found in the 18th century in the creation of the musical dice game. Then only in 1956, the first musical composition "*Illiac Suite*" was created by an automatic computer (Nierhaus, 2009). Nowadays, several projects around the world are emerging inside well-known companies, which are investing in researches that study different mathematical models, sophisticated algorithms, and machine learning methods for automatic generation of music. The project Magenta from Google explores areas of opportunities to automate the aspects of creativity. Also, the CSL lab in Paris from Sony is researching these kinds of models. Newer companies like the startup Aiva at Luxembourg are making specific research on architectures called generative models, focusing mainly on music composition (AIVA, 2020; Google, 2019; McFee, 2015).

There are advances in developing systems for algorithmic music analysis and synthesis and the extraction of short-term relations of repeated patterns of a style of a composer or a genre in music. However, it is still a challenge to obtain a long-term structure for the poly-instrumental generation of music compositions with a low computational cost. The outline sketch design of a music composition could take about a week and up to three months, depending on the complexity of the project and its requirements. Automating this process reduces the costs of production by going from months of development into minutes for a

prototype. In the movie industry and the game development pipeline, music composition is a fundamental part of the project. For this purpose, the ability to speed up the machine music creation increases the productivity of multidisciplinary projects like the ones mentioned previously. Therefore, now a computer can also be an expression tool and not just an automaton of orders.

The most common file structure used to make music analysis is the Music Instrument Digital Interface (MIDI) file, because of its availability on the internet and its support on modern software and hardware. It is also used in music composition and rapid fast prototyping. That is why we focus on analyzing the structure of the MIDI and use it for feature analysis and new data generation through the proposed methodology.

There are several attempts to extract and analyze the existing MIDI files. The problem arises at trying to use powerful deep learning or fuzzy intelligent systems to abstract knowledge from all the unlabeled data. There are not enough available structured or labeled datasets to analyze or to train supervised or semi-supervised models. There are no metrics or enough standard samples of the different aspects of music talking about melody, harmony, and rhythm. Different approaches for music data compression had been applied to label these databases and index the files by segmenting the MIDI messages analyzing the short and long-term structures.

To provide new solutions for extraction, analysis, visualization, and generation of music compositions, the proposed methodology will be capable of extracting the music features automatically from data files by abstracting the MIDI data messages defined by music theory and reflected in the MIDI file structure. These

extracted features are used as knowledge rules to generate new musical pieces applying original algorithms of harmony and rhythm recombination. A critical goal in the proposed methodology is to apply evolutionary algorithms taking advantage of the artificial intelligent processes when a global optimum or near-solution optima is required to obtain in shorter times. In this way, the used algorithms will provide more adaptable and consistent results invariant to lengths and complexity of MIDI files as well as the polyphonic music with different instruments, and variable styles or genres can be created in a reasonable time like minutes or even lower. This achievement with genetic algorithms in automatic music composition can be considered a scientific impact of the proposed solutions providing a new form for creating a music creator.

The results of this research also have a social impact helping people, which are not necessary are experts or professional composers, with their musical composition tasks of experimentation by rapid prototyping music pieces in a short time period. Nowadays, the automatic music generation has a significant economic impact and take about 10-15% of production budget in projects that develop feature movies, videogames, entertainment applications, commercials, TV, etc.(ThinkSync Music, 2020). Therefore, several companies will be able to take advantage of the proposed methodology for speeding up their production timeline as well as for the improvement of their competitive commercial strategy that usually relies on team creativity.

1.2 Related Work in Automatic Music Composition

1.2.1 Advances in automatic music composition

This section presents an evaluation of recent advances and relevant solutions in the area of automatic music analysis and generation. Despite the existence of many reports published in the scientific literature about artificial analysis and generation of music, the number of approaches and methods used for the development of intelligent machine music creators is quite limited. Well-known solutions mainly can be subdivided into four groups depending on used approaches for new data generation such as heuristic, stochastic, deep learning, and symbolic methods.

Among a group of *heuristic composition methods*, it is possible to distinguish evolutionary-based algorithms and dynamic programming approach, which successfully have been used for algorithmic music composition. As usual, evolutionary-based approaches exploit three main concepts of algorithmic composition with genetic programming, which are music representation, searching guide heuristic, and evaluation of a good composition (Dreier, 2015), In the proposal of Dreier a piece of music is represented by a sequence of notes in only the C-major key with their corresponding durations. The objective function introduced as the main concept of applied heuristics is just a process of searching music composition that sounds good in all the sets of all possible musical compositions. Therefore, there is not any quantitative evaluation of the new compositions, in which “goodness” depends on human perception. The quantitative evaluation of new music pieces was proposed

by Eigenfeldt, where the rule-based fitness function is used as a set of musical rules, which a composition must obey to be considered "reasonable" (Eigenfeldt, 2016), The better the music composition sticks to these rules, the higher its fitness function will be rated. Unfortunately, the proposal cannot be used for automatic music generation, because running ten musical pieces in each generation for ten generations requires the user intervention for rating one hundred fitness functions. Therefore, the proposed approach requires an active presence of a human performer, because generating an entire musical composition entails the development of a musical form, which is a highly complex task. Better results are obtained by Kunimatsu, in his music composition model with genetic programming and chord progressions due to a more elaborate description of note features (Kunimatsu, 2015).

Particularly, tree representation of notes and duration provide a better evaluation of the goodness of new melody by computing fitness function that is used, such for comparison of the notes in chords and for comparison of entropy of the partial chord progression. The similar approaches for the development of evolutionary music composers can be found in the proposals of Scirea and Kaliakatsos–Papakostas, which main difference from others reports is the use of constrained multiple objective functions trying to reduce the subjectivity of the evaluation of new music compositions (Kaliakatsos–Papakostas, 2012; Scirea, 2016) Finally, the outstanding results have been obtained in Orpheus system proposed by Fukayama, where the proposed algorithm that can automatically generate music piece as an optimal-solution search problem under constraints given by the prosody of the Japanese lyrics. The melody is designed by dynamic programming by combining the rhythm, chord progression, and accompaniment patterns. However, any quantitative metrics applied to evaluate the quality

of generated music, taking into account only the opinions of classical music composers and the general public on the internet (Fukayama, 2010).

Although several solutions based on heuristic composition methods produce quite acceptable and good audible music, they still have some disadvantages. Mainly, they are sufficiently restrictive in terms of expansion of the model and operate under constrains of unique style for which they are designed. The extracted music features sometimes have a straightforward representation using only pitch and duration accent, and as a consequence, the automatic composers seldom operate polyphonic music. Additionally, there are no metrics to approximate a loss function and to extract music features numerically correlated to the perception of melodies, as well as generation of music is not automatic, and the active presence of a user in some generating processes is indispensable. Finally, the evaluation of automatic music creators is done by the subjective opinions of users.

Deep learning composition methods can be subdivided into deep belief networks, convolutional networks, and recurrent networks, among which the recurrent networks are used more frequently for music generation. For example, Huang & Wu reported in their research deep learning is appropriate for music creation. Using long-short-term memory, they demonstrate that a system is able to learn sophisticated music structures(Huang & Wu, 2016). However, their conclusions have been done using evaluation by only 26 volunteers that rated three samples of generated music. The hierarchical recurrent network has been used by Chu for pop music generation, where the proposed system composes a melody by encoding two random variables that represent key and the duration of a note (Chu, 2016) The music generation is provided by long-short-term memory network, which is conditioned to the scales that allow only notes between C3 to C6 can be voiced. Unfortunately, no quantitative

metrics are provided to assess the goodness of generated 30-second music pieces, the quality of which was evaluated by 27 participants in system tests. Very interesting research has been discussed in (Bretan, 2016), where the combination of two methodologies are used for music generation on deep neural networks. First, an autoencoder learns one, two, or four music features from a dataset, and then using a long-short-term memory network, the music measures are predicted. The proposal must include evaluation using the quantitative metric as mean rank, but the quality of generated monophonic pieces at the end must be rated by musicians during listening. Very promising results in the generation of polyphonic compositions can be found in (Lyu, 2015), where the polyphonic model with long-short-term memory and recurrent temporal restricted Boltzmann machine provides better learning if the input music features are normalized by transposing the note sequences into the same tonality. Another advantage of this research is the applying standard MuseData and the JSB Chorales collections that describe the logical content of standard classical repertory in a software-neutral fashion. Similarly, a system proposed by Johnson for generating polyphonic music using tied parallel networks demonstrates that long-short-term memory networks can learn from the collection without translation invariance of the particular dataset as well as they can predict the sequences of notes from the datasets (Johnson, 2017). It is important to note that systems based on deep learning composition methods can generate efficient musical pieces with musically convincing results even to the ears of professionals and with qualitative harmonization (Hadjeres & Pachet, 2016) Deep Belief Network successfully has been used by Bickerman. They exploit unsupervised learning technique to facilitate automated creation of melodic jazz improvisation over chord sequences (Bickerman, 2010) It has been demonstrated the applicability of probabilistic neural network based on restricted Boltzmann machines for music encoding and creation. Finally, in (Grachten & Chacón, 2017), clear

evidence is found that multiple strategies for adaptation of learned features efficiently can be used in creative processes in terms of the amount of training and in terms of their ability to confront with restricted availability of training data.

In the group of Stochastic Composition Methods, the Markov Models and Generative Adversarial Networks are distinguished. Plenty complete system for automated lyrical songwriting application has been proposed by Ackerman, where the creation of new music is provided after application of some steps such as the corpus definition, feature extraction, model building and evaluation, user vocal conversion to music features, and song generation (Ackerman & Loker, 2016). A corpus-based system is fed with short lyrical phrases producing different options of the style in a given corpus. The system can generate features based on the song lyrics from the vocal line. This method uses random forest for training the Markov chain models. The evaluation is based on the correct prediction of the pitch and rhythm found in the test set, and such a train set was not used to train the model. Another relevant research using Hidden Markov Models (HMM) for music generation has been proposed by Makris, who uses the melody voice from a harmonization system to compute the most probable movement for the bass voice by using HMM training process for calculation of four probability values: beginning, ending, transition, and observation probability (Makris, 2015). Similarly, a web-based application found in (Papadopoulos, 2016) can create or suggest musical lead sheets composition by applying two Markov chains that represent the melody and chord sequence but with tight interaction with the user.

Interestingly, playful research is presented in (Scirea, 2015), where the so-called scientific music generator is proposed to generate lyrics and melodies from real-world data, particularly from academic papers. The model considers two Markov chains, one for tone and the other

one for the duration. The chain is represented as a hash-map that has the current note, and their values are situated in another hash-map containing the transitional probabilities for choosing the next pitch note used as genotype. The training is based on evolutionary algorithms instead of expectation-maximization to avoid overfitting of the model. From the solution space of the population, two individuals are selected as well as the crossover is provided by randomly choosing an index from Markov chain and copying the states from one of them until it reaches the index and the rest of the other parent chains. The prediction of a melody in a dataset will be specified by the fitness function of the probability that the Markov chain presents for the next transition from the current state.

Generative Adversarial Nets also can be used in the creative processes of music real-valued data, when a discriminative model to guide the training of the generative model can be considerably benefited. Yu, in their research, proposes modeling the data generator as a stochastic policy in reinforcement learning using sequence generative adversarial nets that bypasses the generator differentiation problem by directly performing gradient policy update. The reinforcement learning reward signal comes from the GAN discriminator, and then it is passed back to the intermediate state-action steps using Monte Carlo search solving in this way a problem of the transferring gradient update from the discriminative model to the generative model (Yu, 2017). Although deep learning and stochastic composition methods have provided a significant breakthrough in the development of systems for automatic music generation, there are still several disadvantages that restrict their full application. There are not standard labeled datasets for to use with a deep learning approach directly for music generation. The reported in the scientific literature systems for algorithmic creation of new data still generate monophonic music, which frequently is not yet in real-time, as well as

there are not standard quantitative metrics for evaluation of how musically convincing new compositions are. Finally, the user, as usual, must be part of the generative process, at least for subjective assessing the goodness of the entire work.

Several symbolic artificial intelligence methods also have been used for music generation, particularly agents, declarative programming, and grammar composition methods are more promising due to their high-level symbolic and human-readable representations of problems, logic, and search. For example, in (Navarro, 2014), a multi-agent system with human interaction is proposed, where the original harmony search algorithm to generate the composition is applied. It starts by choosing an optimization function to consider the harmony memory, which is a matrix with several solution vectors equal to the size of it. Then a selection is made randomly, with two probabilistic considerations: pitch adjustment rate and harmony memory. There are constrain rules to evaluate the selected vector using specific thresholds.

A famous example of applying declarative programming can be found in (Eppe, 2015), where the conceptual blending of jazz chords is provided using the computational invention of cadences and chord progressions. Some rules of cadence in a search space are blended from previous idioms or keys, and the generated chords are controlled through rules from diatonic music theory. The method starts a search process that interleaves the combination of chord specifications with a step-wise generalization process that removes notes, which cause inconsistency. With priority in pitch notes, the blending is evaluated, and the search space is pruned. The mixing is completed with the general chord type algorithm (Cambouropoulos, 2014), that finally, ensures finding transitions between the chords to achieve the cadence. By using grammar-based generative models Quick and Hudak could

generate the harmonic and music metrical structures. Form two proposed to use grammars, one is temporal, where the duration of phrases parameterizes production rules. The second one is a graph grammar, where the parse trees as graphs allow shared nodes (Quick & Hudak, 2013). Authors report promising results, but some dissonances were found in generated pieces because the grammar does not differentiate between music modes.

In more recent paper authors report improvement of their proposal using graph-based learning from a corpus to obtain production probabilities for musical grammars (Quick, 2016) Concluding the analysis of the relevant methods used for music generation, it is important to mention that convincing audible results vary significantly depending on the desired outcome and the level of acceptance by the user. The methods based on the neural network need labeled databases, which still are not available, so, in the end, they cannot create music compositions with complete development and variation. Also, they need hours of training and use of a large amount of data.

On the other hand, the methods that are based on stochastic processes have results that become repetitive without any kind of progress of the music composition. Lastly, the methods based on symbolic manipulation like rule-based ones need a hard-coded generative knowledge-databases, which can take days for data preparation. The reviewed approaches in the area of automatic music generation are resumed in Table 1-1.

Table 1-1. Grouping the relevant methods for music composition

Area	Subareas	
Soft computing	*Heuristic Composition Methods*	
	Evolutionary Based Algorithms	(Dreier, 2015) (Eigenfeldt, 2016) (Kaliakatsos–Papakostas, 2012) (Kunimatsu, 2015), (Plans, 2012) (Scirea, 2015) (Scirea, 2016)
	Dynamic Programming	(Fukayama, 2010)
	Deep Learning Composition Methods	
	Deep Belief Networks	(Bickerman, 2010)
	Convolutional Networks	(Grachten & Chacón, 2017)
	Recurrent Networks	(Huang, 2016) (Chu, 2016) (Bretan, 2016) (Cox, 2010) (Hadjeres, 2016) (Lyu, 2015) (Johnson, 2017)
	Stochastic Composition Methods	
	Markov Models	(Kaliakatsos, 2014) (Kaliakatsos-Papakostas, 2011) (Makris, 2015) (Papadopoulos et al., 2016) (Scirea, 2015) (Ackerman, 2016)
	Generative Adversarial Networks (GANs)	(Yu et al., 2017)
Symbolic AI	Agent Composition Methods	(Eigenfeldt, 2012) (Navarro et al., 2014)
	Declarative Programming Composition Methods	(Eppe, 2015) (Cambouropoulos, 2014)
	Grammar Composition Methods	(Quick, 2013) (Quick, 2016)

As we can observe in Table 1-1 also presented in (Lopez-Rincon, 2018), in the automatic generation of music, the most used methods were based on Markov models and evolutionary algorithms. However, due to generative adversarial networks keep evolving, a new interest in this area keeps increasing. Evolutionary algorithms have been one of the most explored methods, and we are going to use and improve them in the proposed technique for music synthesis.

Given the previous researches and after different study outcomes of relevant approaches of artificial intelligence for automatic music generation, it is possible to propose the generalized methodology for the automatic processes of music generation. Because of the

standard dimensionality and normalization of analyzed music data are not specified yet, the solution of this problem must be the first step of music features extraction and decomposition. Besides the complexity or training time needed in nonlinear approaches, the other two problems are found. They are looping in music development and wandering, where the first refers to repetitions in the automatic composed music, and the second one implies the melody development without any theme or rhythm. Based on findings during the literature review, the application of artificial intelligence approaches to music generation is a very promising direction of modern research. As a result, based on advantages and disadvantages of budding approaches as well as the required solutions needed in the area, a general architecture of a system for music feature extraction, processing and generation is proposed, where expected contributions will include such data analysis as data synthesis with active interaction with user Figure 1-1.

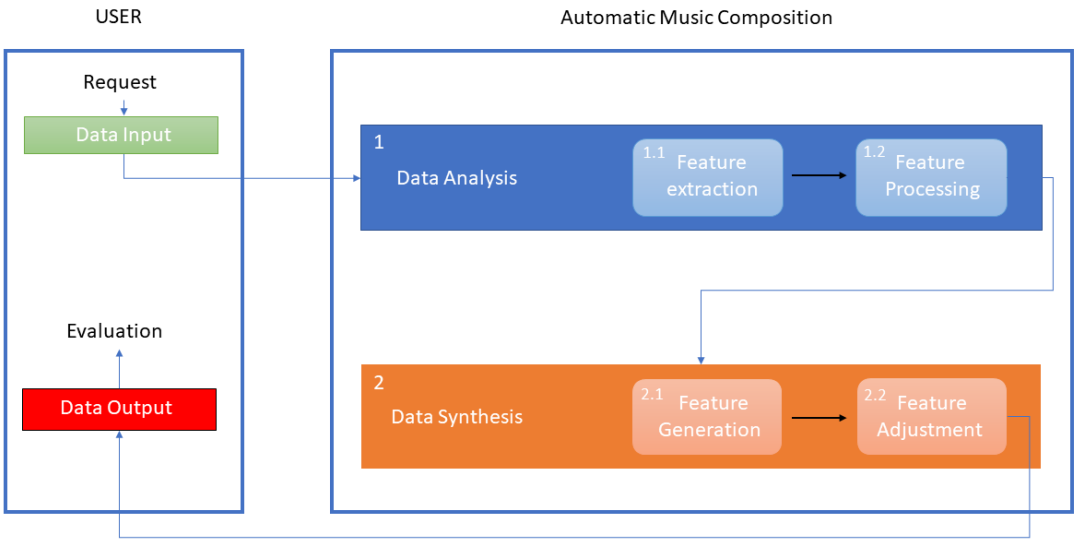


Figure 1-1: General model of an automatic music composition architecture and user interaction.

Due to some considerations based on preliminary experiments, the expected contributions concern advances in the processes for music feature extraction and encoding, taking into account their resolution and temporization by the quarter note instead of the compression rate used in well-known approaches (Herremans & Chew, 2019; Toussaint, 2010). Still, there are not conclusive methods that could be used for the description of different kinds of styles. It is required to apply a preprocessing of data cleaning them by hand. The second expected contribution is related to music synthesis that is based on the previously provided analysis of music features. Particularly, the new non-plagiary harmonic development is going to be constructed from an abstraction of multiple music pieces with a controlled selection of rhythm. The performance of the proposed techniques for music composition analysis and generation are assessed to show their efficiency and competitiveness.

1.2.2 Recent approaches for music analysis and synthesis

Previously carried out a review of recent approaches and prominent applications that have been used for selection of the most prominent directions in the development of highly efficient systems for automatic music analysis and generation. In this section, the detailed technical evaluations of methods for music feature extraction, processing, encoding, compression, and visualization are discussed with a principal objective to justify the proposals derived from the research.

There are several general methods for data analysis, which can be helpful in exploring music data analysis and generation. During music feature processing and encoding, there is a necessity to apply compression methods used for dimensionality reduction of complex data as well as for projection and exploring the behavior of data visually in different environments.

Another use of a new projection of data is the elimination of redundancy in information, which helps intelligent systems to learn and find patterns (Kaski & Peltonen, 2011). There are some primary methods for dimensionality reduction; t-distributed stochastic neighbor embedding (t-SNE) (Brunner, 2018; Maaten, 2014), the autoencoder (Hinton, 2006; Lopez-Rincon, 2018), and specific for music analysis, the spiral array approach for computational modeling of tonality (Chew & Chen, 2005).

The method of t-SNE exploits a clusterization approach in high dimensional data using principal component analysis and neural networks, and it computes thresholds to determine the closeness of the nodes between each other to classify them as *far* or *close*. The nodes determined as *far* are pushed away in clusters to project the clusterization behavior, and authors apply the method of PCA (principal component analysis) to reduce the dimensional redundancy (Hinton, 2006). The approach performs the lower projection by fitting the nodes given in the classification that are *close* within each other to form clusters. However, the principal disadvantage of this method is reducing only redundancy found by PCA and classified by using predefined thresholds. The method is not capable of changing the metrics of closeness other than the one given by the PCA. Thus, it is not directly applicable to music analysis as well.

The spiral array model based on the Tonnetz theory is explicitly designed for music visualization (Cohn, 1998). This theory consists of mapping and modeling the structures like chords, keys, and tension between moments of the composition. Each tone (*C, C#, D, D#, E, F, F#, G, G#, A, A#, B*) is represented by a 3-dimensional point inside the spiral array. The method splits the composition in overlapped windows of time and averages the values of the assigned points to determine the position inside the spiral. Unfortunately, the approach only considers major and minor key signatures, and it uses a fixed window for any composition. Also, it is not entirely adaptable to different keys or changes of speed like beats per minute.

The autoencoder is one of the unsupervised learning techniques on neural networks used to compress data. It contains a hidden layer known as the bottleneck, which connects two parts of a neural network called encoder and decoder. They are layers of neural networks that can be fully connected, convolutional layers, or variations of them. Although it can be used as a deep-learning approach for reducing dimensionality, the compressor does not provide better performance than standard compression algorithms like JPEG (Theis, 2017). The autoencoder is used for specific tasks such as cleansing data, for example, denoising or gap-filling in images using its more sophisticated versions (Liu., 2017). The variational autoencoder (VAE) is a version of autoencoder, where the bottleneck learns a statistical representation of the data (Kingma, 2013). This method encodes into a representation called latent-space, which represents the lower dimensionality projection. The area described by latent space allows us to select a point in continuous 2D or 3D space, which is inside a region from the cluster of features corresponding to higher dimensionality. The decoder also can morph the features between the points of the learned examples.

In the area of visualizing symbolic music, there are other methods based on compression. They are used mainly for reducing repeated figures during the visualization of the data extracted from music pieces. One of them is a compression method COSIATEC (Meredith, 2014) exploited in pattern recognition. It provides finding shapes formed by onsets of the notes. The patterns can have several sizes of shapes and shapes inside the shapes. The algorithm selects the shapes by compressing several times the sequences and then selecting the one with the higher compression.

Another representation based on the Tonnetz theory (Cohn, 1998) is the Isochords approach (Bergstrom, 2007), where authors try to represent the dissonance and consonance of the structures of music in two-dimensional isometric triangles, which describes relations between triads of notes. The next approach for visualization of raw audio using the Self Similarity Matrix (SSM) is introduced by authors in (Foote, 2001). The visualization shows the autocorrelation of segments in time windows manually selected by the user. Following this research, we find the approach of the MIDIVIS (Wolkowicz, 2009), where authors use colors for establishing the relationships of unigrams into fixed-size time windows. The structure of pitch and their length is projected in two dimensions, which can be visualized as patterns in a self-similarity matrix. Since the instruments are dimensional descriptors, and the user can decide which instrument to visualize. The mentioned approach is one of the frequently used music visualization tools; however, its disadvantages are the fixed-size time window and a limited number of colors for pattern representation that might lead to overlapping similarities.

The ImproViz technique (Snydal, 2005) also uses a fixed width of the window, processes six points defined in six beats of the composition, and plots a point at every half value of a

beat. Then the method draws a connecting line between points to represent a shape. In the harmonic graphing, it uses an approach based on the musical genre features, which aimed particularly for jazz. Another interactive tool is a Particle system proposed by Fonteles, to visualize the tones with emitters that are placed in 3D space representing an instrument (Fonteles, 2013). So, the volume is represented by the size of the particle and pitch by the color of the particle.

A visualization technique of music features (like tension) is presented by the Tension Ribbons method in (Herremans, 2016). In this method, the Spiral Array is used to compute different metrics from music composition based on the three-dimensional space. It averages notes found in a fixed window and compares the distances from one window to the next one. This distance is considered the tension that changed in time and used to find the key of a composition. Also, in (Sapp, 2001), a tonal visualizer has been designed to visualize the possible keys or interactions with different moments of a composition by combining different resolutions of time windows. A visualization of the notes with a color map assigned by the circle of fifths is discussed in (Ciuha, 2010). To find the hue-color of note, first, it is mapped by a vector at the center of the circle of fifths, then it is superimposed over the representing tone by vector direction. At the 3-dimensional projection, the length of the vector is proportional to the loudest tone in the given pitch class. When several notes occur at the same time, color is computed by vector addition. So, the resulting vector direction determines the hue, and its length defines the normalized loudness.

In affective computing, there exists an entirely subjective method of visualizing music compositions by selecting photos inspired by listening to a piece (Chen, 2008). This visualization is done using previously gathered datasets from existing pictures with their

emotional description to train a model and to recognize the emotional class in new pictures by matching features found in music such as loudness and spectrum. However, the approach is based on raw signal processing and cannot be directly applied to the visualization of symbolic music compositions. In (He, 2017) authors to visualize the style development and period of music as well as their preference by processing only the tonal key of different compositions.

Despite the newest advances in convolutional and recurrent neural networks in machine learning known as deep learning, it is still an unknown area to adapt them to generate sequences of music data with global and local features. They can nest a hierarchical concept of the information through the connections. These connections are visible through graphed weights between their layers (McCaig, 2016) A *generative model* (*GM*) is used to create new information in a specific domain using given specific parameters, instead of classifying data, which have been the main objective of the first machine learning models.

Even with these new techniques for visualization and generalization of the data in deep learning, they work well exclusively for spatial analysis correlation, and their main goal is auto labeling. We still need a method that allows us to update the knowledge database of a system or even extend it to be able to reverse these layers and use them for generation. These methods should not only allow the classification of independent variables but also to use the relationships between them. There is a paradigm of *rule-based systems with machine learning*, which provides a framework to automate rule extraction (Hailesilassie, 2016). These systems also allow us to provide part of the knowledge database of the system in this computer model in advance and update these rules. This update can be done through interaction with the system or by itself by interacting with its environment.

A *database knowledge* (DBK) in the context of a rule-based system with machine learning, has relationships between *if-then* rules that can expand and infer new relations or decision rules to manipulate the data to perform correctly at a given moment. Contrary to soft computing, where most of the information is not visible, interpreted, or updated along with use. Neither a traditional rule-based system, which is the equivalent to design an expert system with possible decisions and actions hardcoded, has all necessary rules and models in database knowledge (Livingstone, 2010).

As part of artificial intelligence approaches, a sub-area of generative art is the one that comprehends the computer-generated music (Boden, 2009). When a composition is being made, the unique goal of the composer is to express something (Livingstone, 2010). The user would have this goal in mind when selecting the reference pieces in the dataset. By selecting a subset of data, the system could extract features and then replicate those characteristics found in the given information in new music compositions. A rule-based system can perform and finish its goals with a minimum of data that a user could handpick within a determined range, starting from half an hour approximately. This interaction also useful when the selected references are a small number of examples, which is one of the constraints. So, deep learning methods only work with large datasets and require a considerable amount of resources (O'Mahony, 2020).

The analysis of the harmony or melody contour can be studied to find the features of the music pieces (Prince, 2014). Still, very few tools had been developed to study them as a numeric value (Aloupis, 2006). The nonlinear methods in deep learning will also need a large dataset to determine relationships between sets of harmonies to be able to imitate them. These methods need hand-labeled data to generate new data that behaves like a contour or any other

property or aspect in music theory. Other methods with feature extraction are Markov Models and N-grams(Pachet, 2011). After extracting the occurrences of each transition between notes and generating a new sequence given by a random walk through the model, it starts looping. Finally, one representative nonlinear approach that analyzes how music can be viewed by its entropy content is discussed in (Febres, 2017). Authors propose to extract features found in the style, genre, and composer encoded by binary symbols inside music files, and then they are used as a raw signal on the input. Selected symbols were analyzed to find their entropy and frequency, determining their importance to be used as a descriptor of the particular musical composition.

In Table 1.2, the classification of relevant linear and nonlinear music feature extraction, processing, and visualization approaches are resumed concerning symbolic (MIDI) or raw audio inputs (Lopez-Rincon, 2019).

Table 1-2: Classification of music analysis and visualization methods with different input data

		Music Approaches	
		Linear	*Nonlinear*
Input Type	*Symbolic*	(Kaski, 2011) (Chew, 2005) (Cohn, 1998) (Meredith, 2014) (Bergstrom, 2007) (Snydal, 2005) (Fonteles, 2013) (Herremans, 2016) (Sapp, 2001) (Herremans, 2019) (Lopez, 2018) (Ciuha, 2010) (He, 2017)	(Kingma, 2013) (Hinton, 2006) (Lopez, 2018) (Maaten, 2014) (Brunner, 2018) (Sapp, 2001)
	Raw	(Kaski, 2011) (Foote, 2001) (Wolkowicz, 2009) (Sapp, 2001)	(Kingma, 2013) (Hinton, 2006) (Maaten, 2014) (Brunner, 2018) (Sapp, 2001) (Chen, 2008) (Febres, 2017)

The performance of analyzed approaches directly depends on accuracy, complexity, and speed of retrieval of multiple features found in the input data. Linear methods are faster than nonlinear as well as processing symbolic data is faster than raw signal data. Recent methods

mostly are based on algorithms that rely on neural networks or linear algorithms with a specific sequence of steps. None of them is designed to test different metrics to analyze the quality of the resulted visualization process. Unfortunately, most reported approaches do not provide any quantitative computational oriented evaluation of the music analysis and visualization performance. They cannot be used for comparison of efficiency with the proposed method. Only authors of the autoencoder based on t-SNE and Spiral Array try to introduce quantitative evaluation of the music visualization process (Hinton, 2006; Lopez, 2018; Maaten, 2014; Brunne, 2018; Kingmag, 2013). Because of the autoencoder (Hinton, 2006) is the state-of-the-art frequently referenced method; therefore, it also will be used for performance analysis of the proposed approaches.

1.3 Objectives of the Thesis

It is essential first to find solutions to extract, normalize, and encode the music data using a new proposed model, which be able to have a structure capable of generating new data with features locally and globally. Based on the previous analysis of related work, it is possible to formulate a hypothesis that it is feasible to design a model capable of automatically analyze music data, extract local and global features and generate new music pieces by simulating human creativity. After analyzing the features found in data, a system could generate new music composition with no human interaction during the creative machine process. In other words, interaction can be part of the system, but only when it is not part of the process of analysis and generation of the music composition.

The main goal of this thesis is to design a methodology for the symbolic abstraction of MIDI music non-labeled data used for feature extraction, processing, encoding, and visualization, which results then will be applied to automatic synthesis of new non-plagiary music composition.

The originality of this work consists in symbolic representation to make the abstraction of the musical features by considering the *quarter note (crotchet)*, which is the main descriptor of the velocity in music composition because it represents the time unit in the music piece and helps to describe the rhythm of the melody. The crotchet also allows the system to quantify the different combinations of MIDI messages in subsets that represent atoms or words of a specially designed dictionary of harmonies. Due to music can be described by its harmony and rhythm, the original proposal consists in the extraction of their features (pitch, octave, onset, and duration) and encoding them as combinations of words in feature dictionaries. These dictionaries by quantized values describe all the different combinations of note features by hash signatures to be used to find all the occurrences in the composition. With hash tables of the combinations of words from the dictionaries, the model can map out the different sections and repetitions developed in time. This map describes a function that allows the reconstruction of the music piece and analyzes the atomic descriptors, which are given by the different features found inside a quarter value of time measured in ticks. Additionally, a method to visualize and test different music features by quantitative metrics is also provided.

The results of this research can help to determine new architectures used for the analysis of data and to find relevant features as the atomic values of information to abstract them. By achieving these objectives, we are a step forward into the automatic generation of data and

settle the basis for new creative agents to help society move beyond actual expectations of imagination. It will evolve into new possible businesses and the next generation of entertainment and new ways to satisfy curiosity by allowing a broader community to experiment, create, and invent.

1.4 Outline of the thesis

This thesis is structured as follows:

Chapter 2 describes the background of music theory as well as the MIDI file structure messages. The important use of the value of the quarter note to normalize the quantized length of the file is also discussed in this chapter.

Chapter 3 describes the different steps of the 1) methodology to analyze a MIDI file and how it should be normalized so that the length of the MIDI file is self-abstracted by its descriptors. 2) The analysis of the MIDI file after quantizing the messages (events) and how to divide them to get the required features are introduced. 3) With these divisions of the extracted features, we proceed to construct dictionaries based on the information retrieved from the MIDI file. We cover the different parameters from the file structures, and we use these descriptors to analyze the music file visually.

Chapter 4 explains the methodology of data generation with techniques of artificial intelligence implemented by music feature dictionary replacement. Based on the proposed methodology, a new musical piece is generated.

Chapter 5 describes the evaluation of the experiment carried out by applying the proposed approach for note feature extraction, encoding, and visualization.

Chapter 6 exposes the evaluation of provided tests of the proposed methodology for music data generation and user interaction.

Chapter 7 presents the conclusions and a discussion of future work for the extension of the proposed methodology.

Chapter 2 Music Theoretical Background

The Music Instrument Digital Interface (MIDI) protocol is the main file data structure in music composition. This file structure is used whatever of the music composition is done by a human composer or a computer model. We present the structure of the MIDI protocol and the basic music theory used in the proposed methodologies presented in Chapter 3 and Chapter 4. The explanation will extend up to the sound properties that are described through the MIDI and the symbolic parallel representation with the music sheet. This chapter also introduces the concepts of symbolic dictionaries and the descriptions extracted as a compression strategy, which is related to the methodology and the final solution of the analysis of a MIDI file. Finally, the description of a generative model is explained, and the differences with classification models like deep learning are pointed out.

2.1 Introduction

The MIDI file structure is simple and was introduced in August 1983 (Logan, 2020), and it had the first protocol capable of communicating different electric instruments between each other. It is simple and has been supported since then, in both hardware and software. From that date until this long time, its support created a large amount of data that is now free and available on the internet. There are other standards in music like Gpx or MusicXml with a specific use and specialized software. Still, the preprocessing needed to use them as an

ordered file structure to create a database like in (Ghisi, 2017) would have to do notations by hand, and so all the available data will render useless.

2.2 Music Theory

In this section, we describe the concepts of music theory that are described in the MIDI file structure, and that enables the analysis of data for feature extraction, analysis, and generation of new information.

Music notation has symbols that describe the relationships between notes. A note is a sound with the properties of pitch and duration. A representation of a note is called a *quarter note* or *crotchet*, which is a filled-in oval called note head and a straight stem, as seen in Figure 2-1. Its duration is a quarter of the value of a whole note.

Figure 2-1: The quarter note symbol

The quarter note is most of the time, the beat (the unit of time) of the musical piece. But the beat is represented by the time signature of the piece. Time signature describes pulses contained in a measure at the beginning of the piece, or every time the duration of the measure changes, and it is represented with numerals. These descriptions of time signature and quarter value in a measure are represented in Figure 2-2.

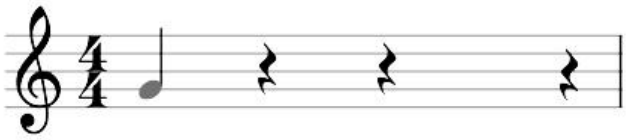


Figure 2-2: A staff with a 4 by 4-time signature with a quarter G note.

Music is divided into specified time-lengths of the same duration. These subsets are called bars or measures. The pulses description inside the measure durations is described by the *time signature,* which we can find at the beginning of a music sheet or any moment where speed is changed. In Figure 2-3, we can see different examples of *time signatures*.

Figure 2-3: Different examples of Time Signatures

We consider the concept of music as the organization of sounds through a determined space of time. There are three primary descriptors of music considered in this thesis. *Harmony* is a set of sounds happening at the same moment. In Figure 2-4, we can see a staff with three-quarter notes that are described to happen at the same time inside the first measure, followed by three empty measures.

Figure 2-4: A staff with the notes C, E, G.

Melody is the set of sounds that are described as a sequence at different moments one after another in an organized way. The notes can have different durations. In Figure 2-5, we can see an example of a simple melody of three-quarter notes of G, C, E.

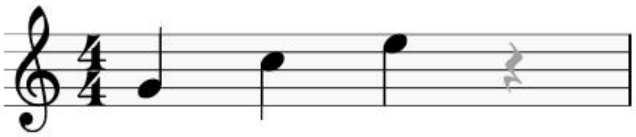


Figure 2-5: Example of three-quarter notes in a measure and a silence symbol of the same duration.

The duration of each note and the moment they begin in time is described as *rhythm*. In Figure 2-6, there is an example of the rhythmic description in the percussion of a song.

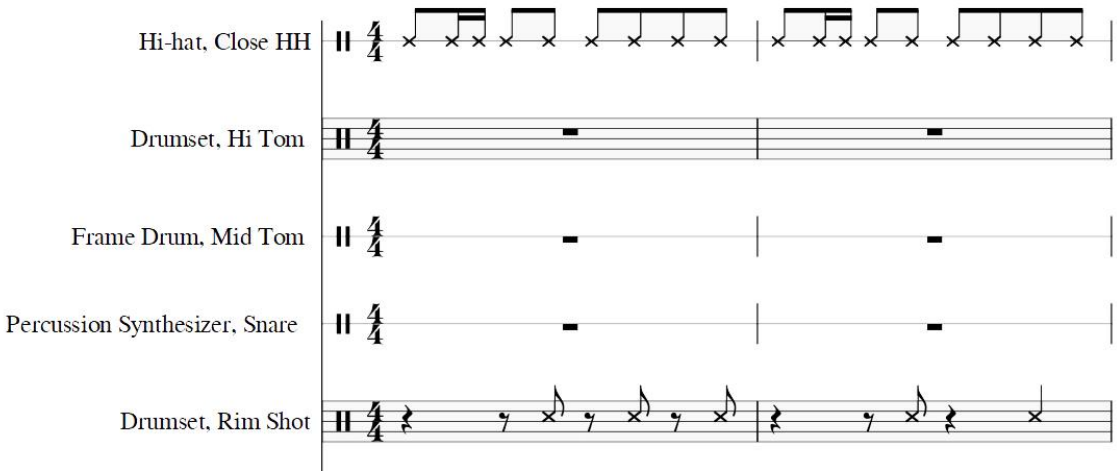


Figure 2-6: Music sheet example for drums.

Time is described in beats per minute (BPM), and this is the speed of the beats through the music piece, which can change in each moment. Figure 2-7 shows an example of the

description of notes for different instruments with a BPM in a music sheet, the BPM in the music sheet is essential to understand the description of the proposed methodology at the analysis part of the music files.

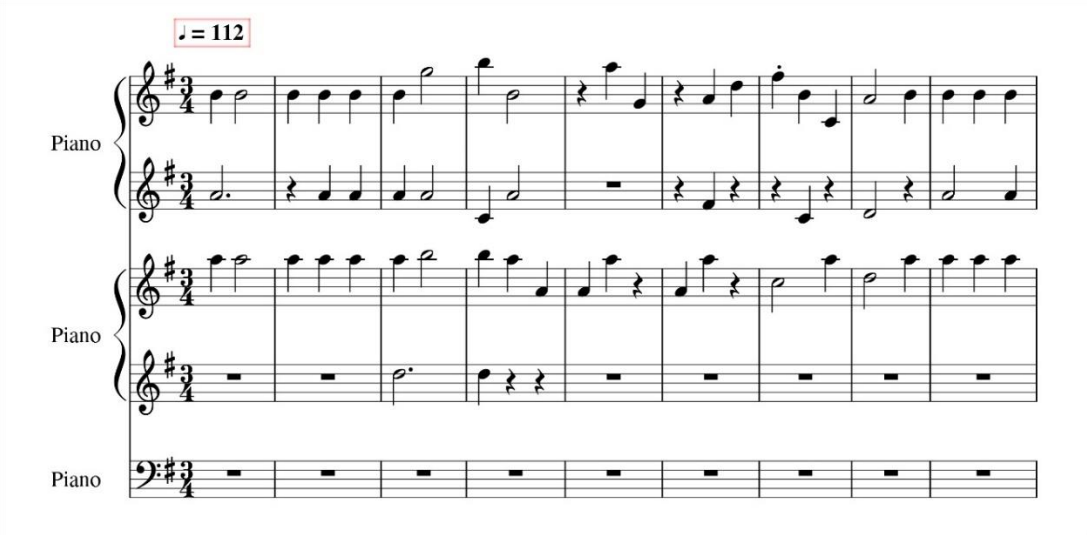


Figure 2-7: A musical piece showing a BPM of 112

In a MIDI file, the BPM is related through the quarter note and the *tempo* value. The tempo value is given in the number of ticks per quarter note, and the quarter note is represented by the number of pulses. The BPM is computed as Eq. 2.1. This equation expresses how many quarter notes happen in a minute. Since we have 1,000,000 microseconds in a second and we compute beats per minute, we have a constant of 60,000,000 divided by the tempo.

$$\text{BPM} = 60{,}000{,}000 \,/\, \text{tempo} \tag{2.1}$$

We encapsulate the description of the melody inside of the harmony. Harmony is the description of two or more notes playing at the same time, and *melody* are notes playing in a sequence. This description helps us to reduce the descriptions of the features of music. With

this reduction, we can start using the structures found in a music file as the only two main features of *harmony* and *rhythm*. With this reduction, we can then consider *rhythm* as the behavior of music in the *x*-axis and the *harmony* as the *y*-axis.

2.3 Music Instrument Digital Interface

The most common file structure to make music analysis is the Music Instrument Digital Interface (MIDI), due to its availability for free on the internet and the support in modern software and hardware. It is also used in every pipeline of music production, and this makes this kind of data available everywhere and compatible between every software and hardware. For these same reasons, it does not have a general standard structure for generation and, therefore, analysis.

The MIDI file is a data structure used to interface with instruments and store the music information events even in these modern days. It is a standard protocol of communications for hardware, and it keeps evolving. New technology adjusts to fit in with the MIDI standard of music composing libraries, hardware, and software. The structure of the messages in this protocol was standardized in 1983, and it is the main structure used in the music industry. Danny Elfman and Hans Zimmer, which are the leading composers of modern film scoring, use MIDI technology for the rapid prototyping of their compositions. With the use of synthesizers in the pipeline production of every film project, they communicate through MIDI protocol with modern scoring computer systems.

The MIDI file format has in its codification events, or messages called notes, which are described as tones (pitches) in a specific moment. These moments are the smallest unit of time called *ticks*. There appear with other events besides the notes that are sound/music descriptors such as instruments, volume, effects, etc. that could occur inside of a piece composition.

Each MIDI pitch could have a value between zero and 127 pitches. We need to identify collections of events with the names *note_on* and *note_off*. A different way to represent the MIDI notes is by associating the volume of *note_on* with velocity value greater than 0 and another *note_on* event with a velocity of *0*, both with the same key number. These events are described with a timestamp or a *delta* value, which is the difference of occurrence compared to the previous event. After filtering the events of *note_on*, they can be transformed into a list of notes. The complete messages of them are described along with their channel, pitch, volume, and duration properties. A typical example of note events inside a MIDI file structure and its transformations into notes is shown in Figure 2-8:

MESSAGES						NOTES			
delta	event	channel	note number	volume		channel	note number	volume	[from : to]
0	On	ch=1	n=58	v=80	} Note 1:	Ch= 1	key= 58	vol= 80	[0 - 239]
239	On	ch=1	n=58	v=0					
1	On	ch=1	n=65	v=80	} Note 2:	Ch= 1	key= 65	vol= 80	[240 - 479]
239	On	ch=1	n=65	v=0					
1	On	ch=1	n=58	v=80	} Note 3:	Ch= 1	key= 58	vol= 80	[480 - 719]
239	On	ch=1	n=58	v=0					
1	On	ch=1	n=65	v=80	} Note 4:	Ch= 1	key= 65	vol= 80	[720 - 959]
239	On	ch=1	n=65	v=0					
1	On	ch=1	n=53	v=80	} Note 5:	Ch= 1	key= 53	vol= 80	[960 - 1199]
239	On	ch=1	n=53	v=0					

Figure 2-8: MIDI events description (left) and conversion into notes (right).

MIDI events carry messages that specify notation of the channel, pitch, velocity (volume), vibrato, panning, and clock signals (which sets the tempo). For example, a

MIDI keyboard or any other controller might trigger a sound-module to generate a specific sound produced by a keyboard amplifier. MIDI data can be transferred via midi cable or recorded to a sequencer to be edited or played back. To be able to play polyphony or more than one instrument at a time, the MIDI file has a structure defined as a *channel.* These are commands to generate or control a *voice* message. A *voice* message like *Note_on* or *Note_off* "passes" through a *channel*, already assigned with an instrument with specific effects like *reverberation*. Instruments and effects can be assigned to a channel at the beginning of the events messages, and all the notes or voice commands sent through this channel will be played with the specified instrument or assigned effect.

There are only 16 possible channels inside a MIDI file, but not all of them need to be assigned or used to make a file work. C*hannel* Nine, when counting channels starting from zero or channel ten if the counting starts at one, is already assigned in all software or hardware descriptions for *percussions*. There is a table of sounds assigned to the MIDI keynotes they belong to a specific sound like, for example, a *snare_drum* is assigned to the MIDI note number 38 or a *hi_hat_foot* which has the MIDI note number 44.

These MIDI events can also be represented as a piano roll illustration, as seen in Figure 2-9. The representation of the events through time, where *x* represents time in *ticks* and *y* represents the *pitch* value of the note extracted from the event messages. These notes are drawn as a filled color box, and length is represented as the width of the box. In this file, we can see the translation of the previous messages into notes for a piano roll representation. When the file contains more than one instrument, they can be separated into tracks by the *channel* number where those events are assigned. We can identify channels or tracks by different colors assigned to them.

The colors can specify different tracks or instruments inside a MIDI file, and here we present the *left* hand in blue color and the *right* hand in green. These colors mean that even when two events are played on a *Piano*, events with the same color are sent to the same channel number, and it separates events of the same instrument. In this example, we can see two different channels divided or represented as different tracks to differentiate the notes played with each hand of the performer. The MIDI file can have more than 16 instruments by changing the assignation before playing the notes and then changing back to the previous ones. This channel changing in execution is the equivalent of having one performer playing different instruments on a piece at different moments.

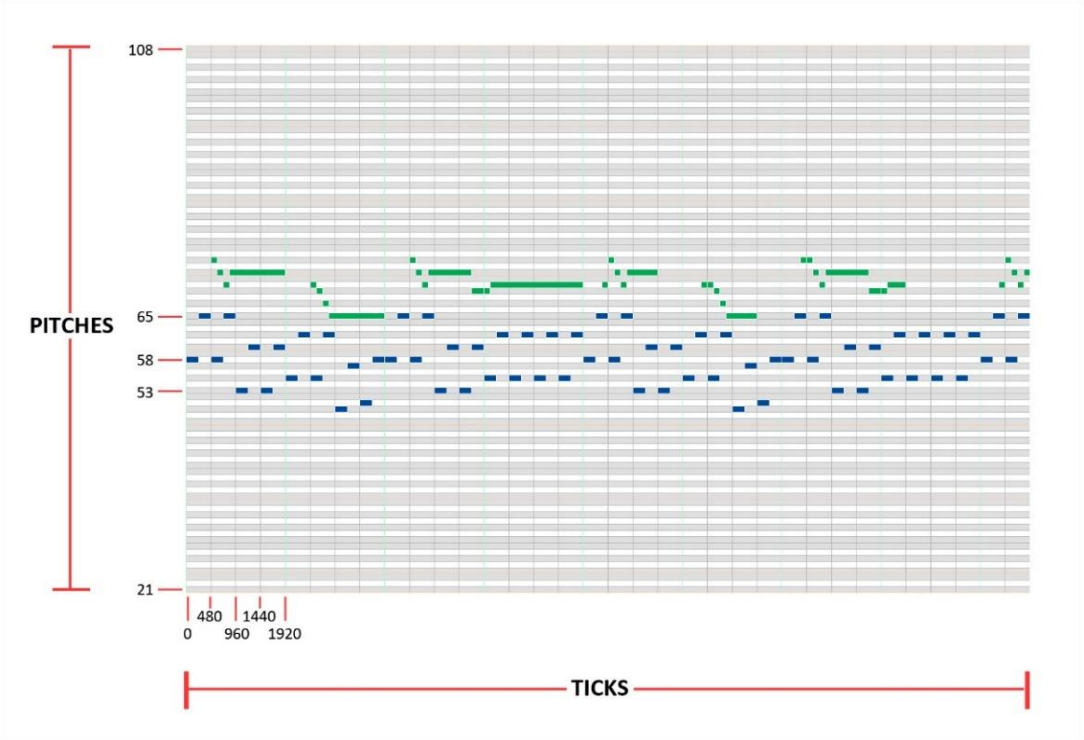


Figure 2-9: Depiction of a Piano Roll of listed notes from the MIDI file where the x-axis represents ticks in time, and the y-axis represents pitch numbers.

This figure also shows the key-number along the axis of the *pitches* as the rows. To illustrate this, we can see the key number 58 with a blue box at the beginning of the *ticks* in time. The same information can be represented as notation in a music sheet from the same MIDI file in Figure 2-10. We can find this representation of the event messages as notes in, where the same Note with key number 58 is in a red frame at the beginning of the notes in the *staff*.

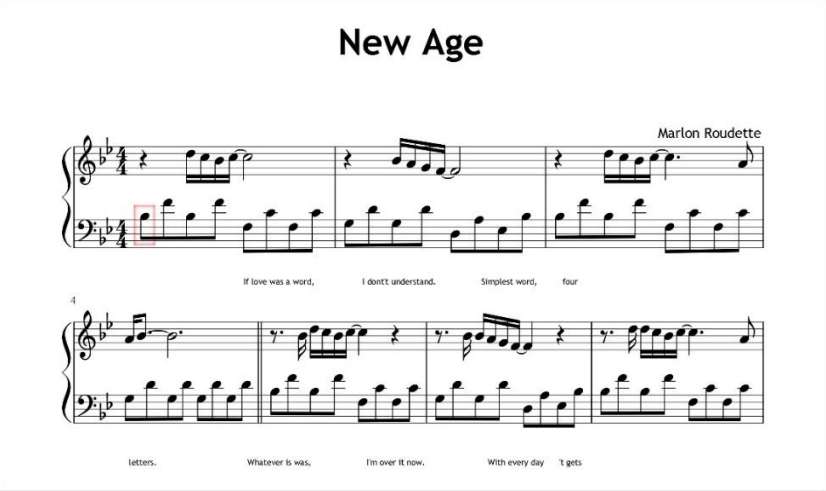


Figure 2-10: Music Sheet of the song New-Age from the artist Marlon Roudette.

In Figure 2-11, we can see two excerpts of the previous file in both representations taken from the MIDI messages transformed into notes. For example, first, we have the MIDI message with a *delta* zero, *event On, channel number* 1, *note number* 58, and a *volume* of 80, then a MIDI message, with a *delta* of 239, the same *event On, channel number* 1 and the same *note number* 58 but with a zero volume. Since we already have that note on the same channel playing, it means that there is a change in the event. When you find the volume zero in a MIDI event of note is the equivalent as a *Note_off* message and we can take then both of

the MIDI messages and transform them into a Note event which starts at the *tick* zero and with a length of 239 starting with a volume of 80 with a *key-number* 58.

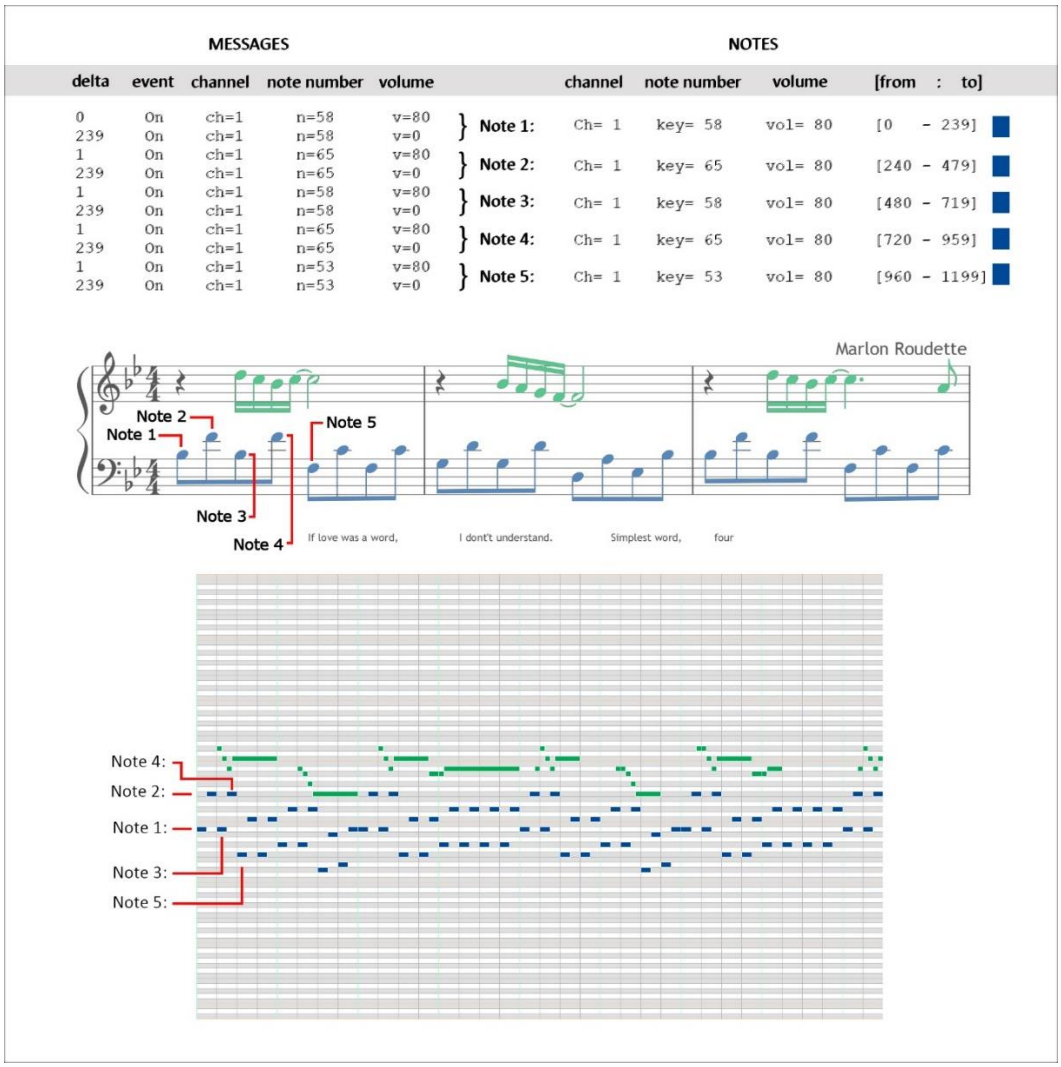


Figure 2-11: Representations of the MIDI messages as music sheet and as a piano roll.

The following messages found with events of *On* are used and paired with their respective events of *Note_off* to transform them into a sequence of **Note** events with their specific *duration*, *volume*, and *key number* played on a specific *channel* that is already assigned to a

specific instrument in the *track.* The MIDI file structure can represent through the event messages dynamics of the sound properties, and the equivalences are shown in Table 2-1.

Table 2-1 Sound properties and the MIDI message related to the property

Property	Description	MIDI correspondence
Tone	16 - 20,000 Hz	Pitch
Loudness	Amplitude	Velocity
Timbre	Parallel frequencies	Wavetables
Duration	Time description	Clock/ BPM
Articulation/envelope	Shape of sound	Vibrato/Attack/After Touch (AT)
Diffusion/spat	Space perception	Panning

These are the basic properties of sound, and the MIDI file has the computational structure to describe them in an abstract representation. In the signal processing market, the commercial software has changed almost only in its graphical interface. The processing of the digital signal is almost the same, and that is another reason why the MIDI file structure is still up to date.

2.4 Discussion

There is 127 sound assigned in the wavetable, which is the indexes of the playable instruments available in the general MIDI hardware. They include guitars, drums, woodwind instruments, piano, among others. There are extensions of the sound tables that are available at specific hardware as synth workstations for production and studio mixes. There are also extensions through virtual instruments that can be created with sound files and assigned

through virtual wav tables of sounds. In this thesis, we are going to consider only the sounds of the general midi with the basic 127 sounds. Most researches considered only one instrument to create the compositions, and the scope of this thesis is enough to probe the concept and hypothesis.

Another consideration of the MIDI file is the one regarding the events. There are control events for dynamics, which are used to increase the render of the final stream signal. We are considering only the events that involve note events. This consideration is due to the scope of the compositions and not the synthesis or rendering of the sound. The music file is done in an essential volume dynamic manipulation to appreciate or hear a representation of the compositions. The composition is the primary purpose of the research, the dynamics are part of the interpretation, which can change from player to player, and it is out of the scope a model of interpretation of the music piece.

Chapter 3 Methodology for feature analysis, extraction, and dimensionality reduction of musical data.

In data analysis, the first step is to determine the feature extraction of data. The selection of the features to be observed by a statistical method, machine learning, or human will directly impact the performance of the results. These apply if the analysis is used in classification or data generation. Cleaning data and selecting features sometimes need an in-depth study of the information, to achieve this, filtering, visualization, and testing tools are needed. Specific details of the structure of the file are described to understand how and why the proposed method works in the way it does. The details of the transformations of the structure found in data are explained. The practical uses of these methods are listed as examples. For a general understanding of the method, a flow chart is presented, and each step is explained later to achieve the results of the testing tool and visualizations.

3.1 Preliminary consideration

The development of techniques for music visualization is an essential and still open problem in the analysis and creation of the quantitative profiles of single or multiple compositions, which could be used as required constraints in music generation or music classification processes. When generating creative data with no fitness function, it is hard to select or to find appropriate measurable features. This methodology normalizes data in MIDI files by 12-dimensional vector descriptors extracted from tonality as well as a novel technique for

dimensionality reduction, and visualization of extracted music data in 3D projections is discussed. The harmonic features in the music composition are found by sliding a non-overlapping window. Then a self-similarity matrix is computed using distance metrics to analyze and project the resulting 3D feature vectors. Three-dimensional projection creates a quantitative profile of a composition, which correlates the tone similarities along with the music piece. The dimensionality reduction is compared with a well-known autoencoder. Conducted tests showed that our method preserves up to 90% of the original data in the projection of a reduced dimension. The advantages of the proposed method consist of a novel technique that provides interactive visualization and dynamically adjusts different metrics to observe the behavior of data during music information retrieval and recognition.

Music visualization is used to read, write, and edit music compositions in the creation context. This kind of visualization can be limited only by the precision of feature selection and the performance of the data extraction tool. Therefore, based on modern techniques of artificial intelligence, data projection, and dimensionality reduction used for downgrading the computational resources are needed for appropriate selection and analysis of specific features found in data. The visualization sometimes needs the flexibility to be used with different quantitative metrics to compare them between each other, when observing their related behavior either based on perception or interpretation of analyzed data.

Contrary to conventional dimensionality reduction methods, it is essential to control selected redundancy due to the nature of music data properties found in tonality, rhythm, speed, etc. That is the reason why traditional methods for data projection do not provide techniques to visualize and understand features, which identify genres, styles, or authors.

The study of tonality and rhythmic descriptors can help organize and find relevant data behavior. Frequently used patterns in data could be used for music classification and automatic music generation due to its quantitative representation precisely measured by specifying metrics of different elements in music composition.

There are numerous unsupervised and semi-supervised machine learning techniques; however, they are not practical to study data from music pieces. To make use of supervised learning methods as deep learning, we need a large amount of labeled data but, making annotations by hand is not so practical. Classification systems like key signature usually are genre bias, and thus, it favors one tonal key over another according to the specific style (Korzeniowski & Widmer, 2018). As a result, these systems only work according to a specific genre for which they were designed.

3.2 Feature extraction from MIDI files

The proposed method for music visualization consists of splitting a MIDI file in quarter note size windows. Then translate the information found in each slice of time into 12-dimensional vectors of harmonic music features. After representing the data in these vectors, they are reduced from their 12-dimensional space into three-dimensional space and projected. This projection is achieved through a self-similarity matrix (SSM). The process steps of the proposed method are depicted in Figure 3-1: Process steps of the proposed method for dimensionality reduction and projection of extracted data from music compositions.

The MIDI file format defines the moments in which the notes of a music piece occur. It has two types of sub-structures called *header chunk* and *track chunk*. The standard MIDI file (*SMF*) consists of one header chunk and one or several track chunks as follows (Back, 2020).

SMF = <HEADER_CHUNK> + <TRACK_CHUNK> + <TRACK_CHUNK> +…

Both types of chunks have a literal identifying string encoded by ASCII of 4 bytes long. Mainly, a string *MThd* is an identifier for a header, and *MTrk* is for track chunk. They indicate that it is a MIDI file. The header chunk is 14 bytes long, and it has the following format.

HEADER_CHUNK =<MThd 4 bytes> <Header_length 4bytes> <MIDI_format 2bytes> <Number_of_tracks 2 bytes> <Time_divisions/quarter_note 2 bytes>

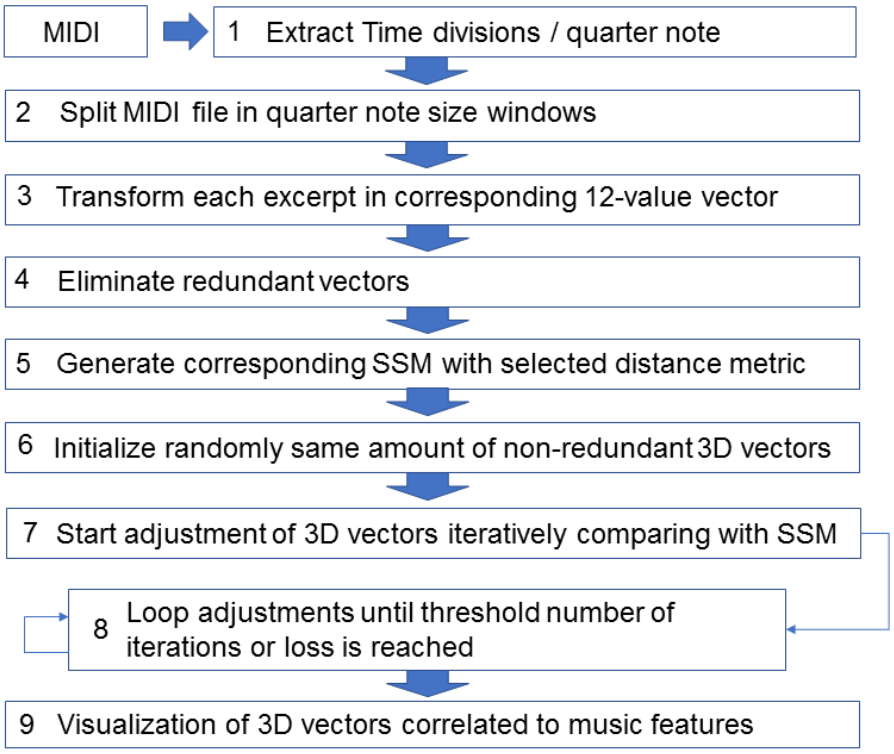


Figure 3-1: Process steps of the proposed method for dimensionality reduction and projection of extracted data from music compositions.

Four bytes located after string *MThd* represent the length of the rest of the MIDI header. The following 2 bytes define MIDI file format labeled as 0, 1 or 2 for single-track file, multiple-track file, or multiple song files, respectively. The next 2 bytes are the number of track chunks appended to the header, and the last two bytes represent the time divisions/quarter note of the MIDI file. The value found in time divisions/quarter notes is used as a window size for splitting the MIDI file. The quarter note value is declared only once, and the rest of the notes (track events) in the file are related to this value.

The track chunk has the following format.

TRACK_CHUNK = <MTrk 4 bytes> <Track_length 4bytes> <Track_event variable> <Track_event variable>…

A track event consists of *v_time* (a delta time since the last event) and one of three types of events: *midi_event* (MIDI channel messages such as note-on or note-off), *meta_event* (SMF meta event) and *sysex_event* (SMF system exclusive event). Each event consists of the descriptions of the music notes, and their properties are used by the proposed methods for the generation of feature vectors. Mainly, the onset, pitch, and duration of each note are the principal elements to be processed.

3.3 MIDI Windowing

The speed of the music piece is described by the *quarter-note* duration in ticks along with the event messages in the file. The *crotchet* or quarter-note is found at the beginning of the MIDI file. The event messages are related to the length of it. The size property in ticks of the

quarter-note is used to compute the splitting profile of a song. According to Figure 3-1: Process steps of the proposed method for dimensionality reduction and projection of extracted data from music compositions. The entire music piece is subdivided into the declared measures inside the MIDI file structure then each measure is again split according to the size of the crotchet. The composition ends up as several pieces coming from the non-overlapping sliding window.

A 12-dimensional vector is filled with the information found in every instant taken by each window. Each vector is correlated with all the different possible tones of equal-tempered music *[C, C#, D, D#, E, F, F#, G, G#, A, A#, B]* and it represents which notes are found in every slice of the composition. The vector has a binary representation, and if a note is present in the window, then the position that corresponds to a tone has a value of one. For example, if a window has a sequence of *C, E, G,* the resulting vector is defined as *[1,0,0,0,1,0,0,1,0,0,0,0]*.

After slicing the complete song into *quarter-notes* size windows, the SSM of distances d_i between all the vectors can be generated. Each vector of the composition is compared to the next one assuming distance as the selected metric. Every extracted vector is inserted into a hash table to find only the different combinations of the harmonic tones. SSM is filled with each of the distances between tonal vectors, where each row is cross related to a column with all the other tonal vectors as shown in (Eq. 1),

$$S = \begin{bmatrix} d_0 & \cdots & d_n \\ \vdots & \cdots & \vdots \\ d_n & \cdots & d_n \end{bmatrix} \tag{1}$$

where *S* corresponds to the SSM, each element is a distance value d_i, and the n is the total number of slices of the composition.

The distances between vectors are computed among each row vector $\hat{X}$ and column $\hat{Y}$ of the tonal vector-matrix found in the composition slices. The similarity distances between two binary vectors are computed using the Hamming distance (H) [29], as shown in (Eq. 2).

$$H = \sum_{k=1}^{n} \left| \hat{X}_k - \hat{Y}_k \right| \tag{2}$$

To reduce dimensionality, we loop through all the vectors and compare the distances to the actual ones in the SSM. In the lower dimension, we take $\hat{V}_i$ and compare the distance to $\hat{V}_j$. Then this value is compared to their corresponding distance in the SSM of the higher dimension according to (Eq. 3),

$$| \, d\left(\hat{V}_i \, , \hat{V}_j \, \right) - D_{ij} \, | > t \tag{3}$$

where $\hat{V}_i$ and $\hat{V}_j$ are the vectors in the lower dimension and D_{ij} used in the equation are the distances in the SSM in 12D space. If the difference is higher than a threshold *t*, we adjust the position of the vector according to (Eq. 4), with a given increment α until the distances are close to the ones in SSM.

$$\hat{V}_i = \left(\alpha \, * \hat{V}_i\right) + ((1 - \alpha) * \, \hat{V}_j \,) \tag{4}$$

The adjustment $a\hat{d}_{ij}$ of the representing vectors is normalized to the higher magnitude. For example, if the Euclidean distance is selected and the vector is a binary 12-dimensional

set, then it is dived by the square root of 12, which is the maximum possible value. The distance of the representation in the lower dimension is also normalized. The total amount of possible combinations in the harmonic space of 12 tones is equal to 2^{12}= 4096. When a composition is analyzed the $a\hat{d}_{ij}$ is scaled to the proportion of all combinations. This scaling is achieved by dividing the total amount of vectors found in the music composition by the constant *4096*.

The adjustment is multiplied by a count of repetitions of the vector divided by the total amount of vectors found in the subdivisions, as shown in (Eq. 5) to visualize and represent the number of computed vectors.

$$a\hat{d}_{ij} = \frac{1}{4096} * \ d(\hat{V}_i\ , \hat{V}_j\ ) * \frac{a_i}{V} \tag{5}$$

where $\hat{V}_i\ , \hat{V}_j$ are two current evaluated vectors, a_i is the total count of the vectors found along with the composition, and V is the total amount of found vectors. We can adjust stop criteria of the dimensionality reduction by setting a specific number of cycles or by using a regression error metric as the mean square error (MSE) (Li & Zhao, 2001a) using (Eq. 6).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(|\ d(\hat{V}_i\ , \hat{V}_j\ ) - D_{ij}|)^2 \tag{6}$$

The subtraction between the original SSM and its new version of reduced dimensionality converges over time.

Another possible stop criterion can be formulated from the adjustment of vectors in 3D space, where a small change between the positions of vectors can be counted and used as a

threshold. The convergence of the projection is achieved when the adjustment is decreased under the previously set threshold. The size of steps in the increment of α value can create a wobbling motion if the adjustment is too high. When this happens, the optimization starts to jump around the threshold, and it could take too many cycles to converge. This looping means that the adjustment needs to be reduced until changes are smooth enough to get the result of the fitness of the projection, and the error is minimized.

3.4 Adjustment Average Error

The softer stop criteria for defining the required number of cycles is based on computing the adjustment average error (AAE) using (Eq. 7), where each adjustment (a_i) in the batch of an epoch is added up and divided by the total amount of epochs (p).

$$AAE = \frac{1}{p}\sum_{i=0}^{n} a_i \tag{7}$$

In many conducted tests, both MSE and AAE, errors have been computed and compared using an empirically defined threshold t set to 0.67, which reached a total count of 300 epochs.

The difference in the behavior of errors is plotted at each epoch using both (Eq. 6) and (Eq. 7) in Figure 3-2: The red line shows the mean square error (MSE), and the green line represents the average adjustment error (AAE) with a decreasing-only behavior. The y-axis represents the percentage, and logarithmic values of errors, and the x-axis represents the number of epochs. The red line shows the mean square error (MSE), and the green line

represents the average adjustment error (AAE) with a decreasing-only behavior. The y-axis represents the percentage and logarithmic values of errors, and the x-axis represents the number of epochs. We can observe how MSE and AAE metrics had different behaviors. The decreasing-only behavior is the best fit to set a threshold and avoid looping until convergence. At the end of the run, when the projection converges, the MSE metric still shows the total error missing from the higher dimension.

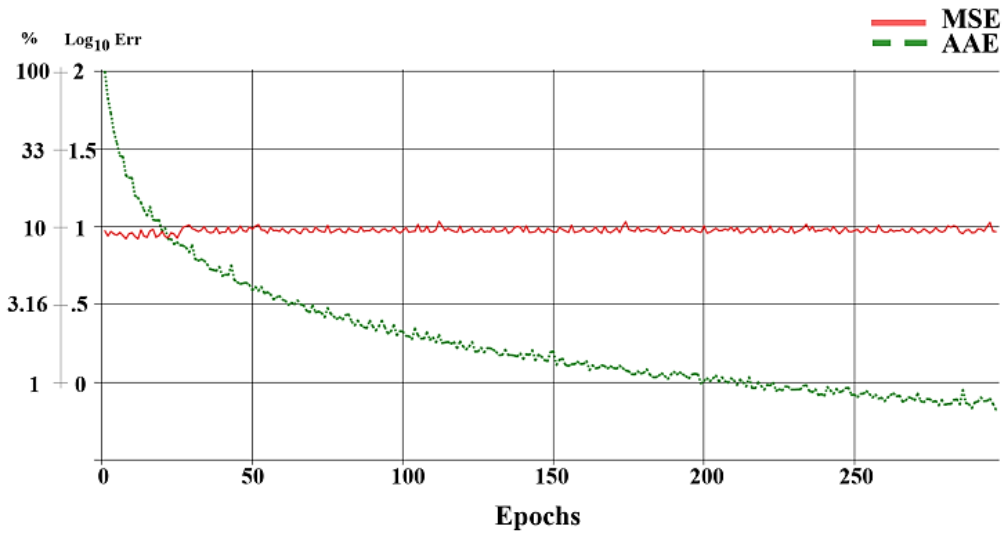


Figure 3-2: The red line shows the mean square error (MSE), and the green line represents the average adjustment error (AAE) with a decreasing-only behavior. The y-axis represents the percentage, and logarithmic values of errors, and the x-axis represents the number of epochs.

From the example where the convergence threshold was set to 0.67, the MSE does not exceed 11.5%, comparing the distances between vectors from the higher dimension reducing it to low dimensionality space. It means that 90% of the original distances of the 12-dimensional tonal vectors are preserved in the 3-dimensional space using MSE in our proposed method. The obtained results presented in Figure 3-2 shows the reduction of the AAE with an increment of the loops for adjustment of 3D vector positions. Using AAE in

our methods, it is possible to select an appropriate threshold depending on the number of epochs. Therefore, AAE is more suitable to use it as a stop criterion of the algorithm for adjustment of vector positions with an error of less than 10% after 20 iterating cycles.

The convergence speed can be increase as follows; the iterative step (see Figure 3-1: Process steps of the proposed method for dimensionality reduction and projection of extracted data from music compositions.) can be optimized by computing the distances of SSM previously. There are only 4096 possible combinations of tones; therefore, all distances between those combinations can be previously pre-computed. In the image presented in Figure 3-3, all distances are normalized between zero and one. They are shown in greyscale, being zero the closest distance represented in black and the longer distances represented with brighter gray.

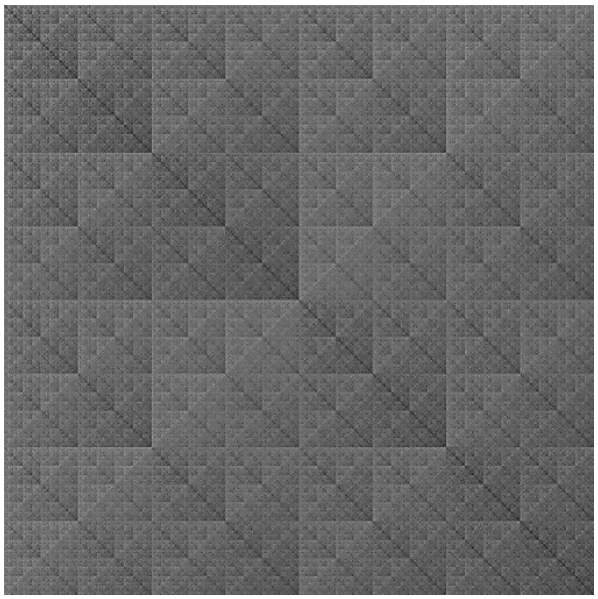


Figure 3-3: The SSM with 4096 combinations of the 12 tones matrix. Each column and each row represent the distance between all of them. It is also a symmetric matrix with a black diagonal, which depicts the distance of each vector to itself with a value equal to zero.

The use of this method over all the possible combinations of the 12 possible combined tones tends to produce a sphere (see Figure 3-4). After some iterations, when a spherical form is approached, the projection collapses, and vectors start to push each other in a loop. The next step was to set the unitary magnitude of each vector for visualization. The result is shown in Figure 3-5. Even with uniform scatter vectors, no information about the notes or the harmony is visible yet.

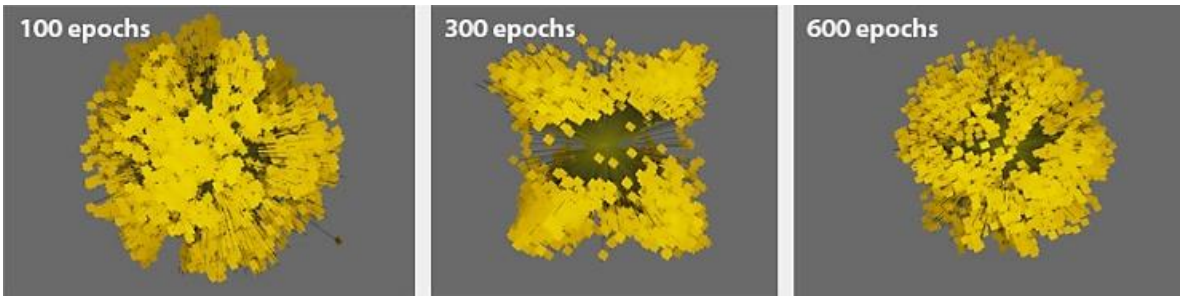


Figure 3-4: Initial 100 epochs (left), 300 epochs (middle), 600 epochs (right), these numbers were selected to show the loop of the 4096 harmonic vectors and their behavior.

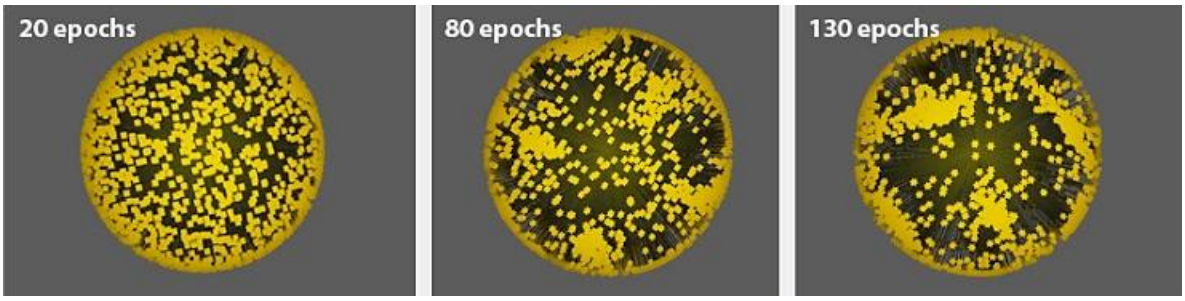


Figure 3-5: SSM with unitary magnitude. Initial 20 epochs (left), 80 epochs (middle), 130 epochs (right).

3.5 Proposal for spherical projection and visualization

This section covers the description of the experiments to evaluate the performance of an extension of the proposed method for feature extraction and dimensionality reduction. The

first set of experiments demonstrates how to visualize harmonic relationships. The second experiment shows clusters of vectors in reduced dimensional space considering collisions between them. The last experiment is used for comparison of the projections of all possible harmonic combinations obtained by the proposed method and the autoencoder as a possible reference. The extension consists of assigning a specific color related to the given 12-value vector from the harmonic window. This given color reveals the relation between the vectors in a piece of music. We show previous results to compare with the extension of the colored vectors and the collision detection to visualize the results.

There are three main experiments and results obtained from the proposed method. The previous method was extended with HUE based coloring, feature collision detection, and a random initialization for spherical visualization were tested. Then, there is the extension of spherical visualization, adding collision detection, and lastly, the final testing of the developed framework. Here the feature extraction and visualization methods are tested.

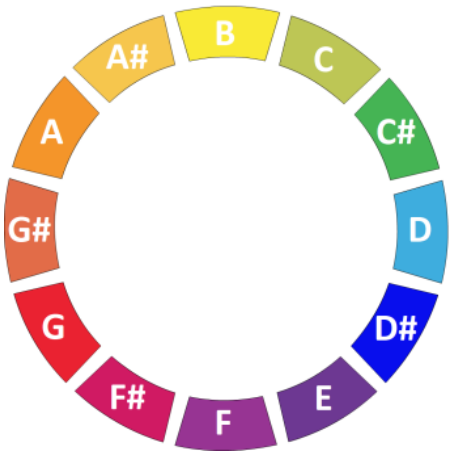
A coloring practice is proposed to find the relation of the vectors according to their harmonic similarity by providing an organized, evenly distributed color to each harmonic value. These colors come from the HUE wheel to get a linear change according to the representation of the tone change. Then the importance of collisions for the dimension reduction in the iterative process is shown by generating clusters of the vectors and the difference between an ordered initialization of the vectors against a random initialization comparing the convergence in iterations.

Finally, an example of the complete normalization of all the possible combinations of harmonic space found in the proposed method is compared with the neural network-based dimensional reduction method, the autoencoder.

3.6 Description of HUE based visualization

A piano is a musical instrument that has seven octaves and represents the complete set of octaves found in tempered music. The MIDI file also has these octaves and three additional ones representing every note within them by values between 0 and 127; they correspond to octaves labeled from -1 to 9 (Back, 2020). In a piano, those values correspond to a subset of MIDI values between 21 and 108. Each of these values in MIDI corresponds to a frequency found at each note of every octave. Tones of equal-tempered music lie in a range of [8.1757989156 - 15.4338531643] in octave labeled as -1 of the MIDI standards. All the following notes in every octave are multiple of these initial values. As a reference, the note *A* in octave 4 is called middle *A*, and it has the frequency value of 440 Hz as well as number 69 is assigned to it in MIDI standard (Stolzenburg, 2015).

The consideration, to assign a color to each of the twelve notes, is to use a discrete division of color tones as it is shown in Figure 3-6. Types of color expressed by hue color space are also split up into 12 incremental subdivisions in a discrete manner and then transformed back to their red, green, and blue values (RGB) assigned to their corresponding index. They were selected in a similar clockwise direction in the same order as the tones found in octaves. If the 12-dimensional vector has more than one value, the resulting color of the vector in three dimensions is computed by averaging tones of the notes found in it. We measure distances between all used 4096 possible combinations of the 12-dimensional vectors and compare them by each other, creating the SSM with Euclidean distance as a principal metric.



INDEX	RGB			3D		
C = 0	254	242	0	0	-1	0
C# = 1	141	199	1	0.9	-0.4	0
D = 2	0	166	82	0.3	-0.4	0.9
D# = 3	1	168	158	-0.7	-0.4	0.5
E = 4	0	84	165	-0.7	-0.4	-0.5
F = 5	46	49	146	0.3	-0.4	-0.9
F# = 6	102	46	145	0.7	0.4	0.5
G = 7	146	39	143	-0.3	0.4	0.9
G# = 8	238	28	37	-0.9	0.4	0
A = 9	243	101	35	-0.3	0.4	-0.9
A# =10	255	148	0	0.7	0.4	-0.5
B =11	254	197	2	0	1	0

Figure 3-6: Hue representation of notes (left), corresponding RGB values with coordinates of 3D positions in an icosphere.

The first set of experiments shown in Figure 3-4 has been repeated to visualize the color labeling of the projected 3D vectors. The expectative was to find similar colors placed closer at the same number of epochs as in previous experiments, but visualizing colors that represent tone combinations with the corresponding hue values (shown in Figure 3-7). This experiment runs without applying normalization of magnitude and without the detection of vector collisions.

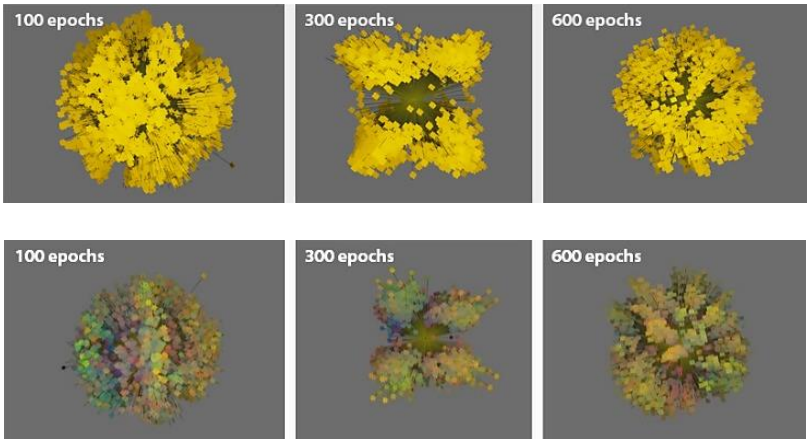


Figure 3-7: SSM with color labeling for Initial 100 epochs (left), 300 epochs (middle), and 600 epochs (right). Upper row, results of the experiment without coloring technique at the bottom, results of implementing HUE visualization at each vector.

3.7 Discussion

Although there is a visual projection by dimensionality reduction and perception of representative harmony vectors, there is still perceptual occlusion. The problem arises when distances on occluded vectors have from the silent vector and adjacent vectors the same values. The magnitude results in the same distance between harmonies and thus results in the occlusion. In Chapter 5, there is an explanation and a collision detection solution to avoid these kinds of missing harmonies projections. It is shown in the experiments how the projection is completed with this algorithm, and different results are shown.

Chapter 4 Methodology for music Data Creation Based on Dictionary Extraction and Feature Recombination

After previous chapters that describe music data analysis and generic nonlinear generators, this section describes the method for Musical Data Creation (MDC) with the developed theory. These approaches present a method to create stems of information extracted from analyzed data that are combined within themselves and by using different sequences and approximations can generate variations at the descriptions of the generated information. Generative Models based on nonlinear architectures can have a high computational complexity, which is reflected in its training along with specialized hardware to implement and test. The model is a simplified architecture of linear analysis of two compositions and recombination with a *biased genetic algorithm*. It can be conditioned to generate specific data depending on the given original files as seeds for the composition. It is a simplified model as an alternative to architectures with higher complexity like Variational Autoencoders or Generative Adversarial Networks. Unlike the mentioned models, this method works with the analysis of two MIDI files and abstracting with an *encoding* method the different features into their main musical components and with a *genetic algorithm* adapts the harmonic development of one of the compositions into the rhythmic section of the other file. It can run on CPU for rapid prototyping and testing results; every iteration of the algorithms generates a mixture of the features with variances in every generation of the harmonic adaptation. It has the advantage of generating a complete musical piece (which depends on the rhythmic section), and it can have different sections and changes in the harmonic development. The adaptation error rate is below the 1% measured with a *mean*

square error on the abstraction of the features. The following introduction shows the motivation to develop a meta-heuristic adaptative method and not a trainable one based on nonlinear architecture like a neural network. Then the representation and definitions of music features in MIDI files are described, and the general process steps diagram, and finally, the process steps definitions.

4.1 Preliminary considerations

Automatic music composition with artificial intelligence can be used in movies and videogames. Also, the procedures of the different steps are automatic or interactive as a compositional tool or as a creativity agent. Most of these methods can only assist with parts of the composition process, not as a whole. Therefore, combining symbolical representations, compression techniques, and artificial intelligence as genetic algorithms are needed for appropriate feature selection, analysis, and recombination for new data generation that complies with different needed characteristics at the final production.

Contrary to conventional methods for music composition based on procedural or artificial intelligence methods, sometimes a user could need different results with variations of the same parameters as complete templates to start working on them as structures. These results can then be completed with audio textures for the production of music composition. The composition step of a music piece is only part of the job to have a completed, usable material to use. After the music sheet is finished, the process of recording, editing, and mixing is part

of the production of the piece, which will render the final step of the composition also to use it as a playable piece for live performance, television, movies, or videogames.

There are different techniques for unsupervised, semi-supervised, and supervised machine learning methods, which are not practical to analyze music data or to train data to generate new music composition files. Deep learning requires a large amount of data, and making annotations by hand is not practical. In models for tonal-key classification like the proposed in (Korzeniowski & Widmer, 2018), usually are genre bias, and training has to be done each time a specific style is desired to classify.

On the other hand, Artificial intelligence is mostly known in areas of data analysis for classification. Generative models otherwise are suitable for the generation (synthesis), completion, or even removal of data. A recent application in image processing, which combines both classification and generation, is image captioning. One advantage of automatic image captioning is indexing, which is essential in content-based image retrieval (Hossain, 2019). Supervised methods learn on labeled datasets, which takes time to process, clean, and test. The datasets context should be bias-free to work correctly and to train a model which can be crucial in domains like criminal justice, infrastructure, or finance (Gebru, 2018) or even in clinical aspects (KS & Sangeetha, 2019). There are labeled maps for road planning, which are satellite pictures with labels that indicate land use, in which such labels had to be prepared (Helber, 2017). Generative models can help to expand datasets or generate new examples from the abstraction of features from different classes.

4.2 Feature definitions of music in MIDI files

Music in a MIDI file can be described as ordered sounds with three main features: *rhythm*, *melody*, and *harmony*. *Harmony* is described as two or more notes played at the same moment, and *melody* is described as a series of notes played at different moments so that we can describe harmony as a melody with an offset of the notes with an equal to zero value. A MIDI note is equal to an integer value related to a key on a piano, which ranges from 0 to 88 and produces a sound at a specific given frequency each of them. *Rhythm* is developed when a note starts playing, and it is called the *onset.* Another feature is the *duration* of a MIDI note, and these sub-features can be described in a hierarchy of the different features of music, as depicted in Figure 4-1.

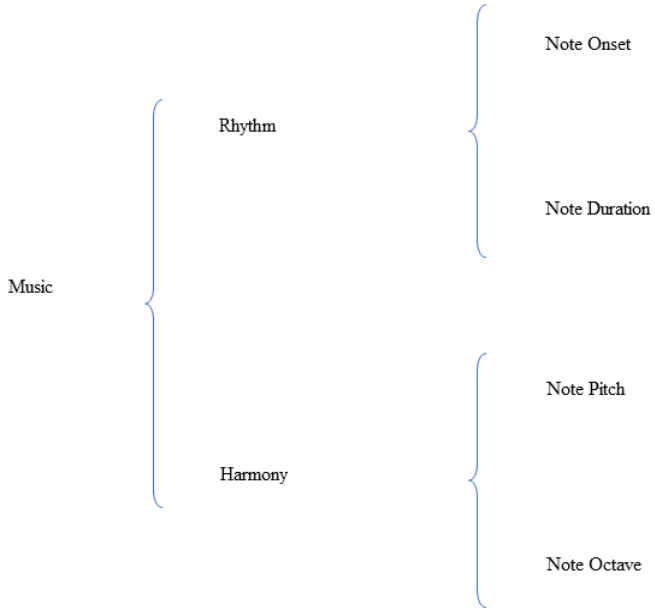


Figure 4-1: Hierarchy of structure of music features found in a MIDI file. Through events related in time by ticks, music can be divided as features of rhythm and harmony, which at the same time can be subdivided into sub-features of onset, duration, pitch, and octave.

With these music features and sub-features in the hierarchy, a note can be described as a structure of four integer values and define *composition* as a sequence of notes as events to be played by instruments. These music features are found in the structure of MIDI files, and they work by describing *messages(events)* as playing notes to generate ordered sounds. The goal of music data generation is done by mixing these features of two different MIDI files, as depicted in Figure 4-2.

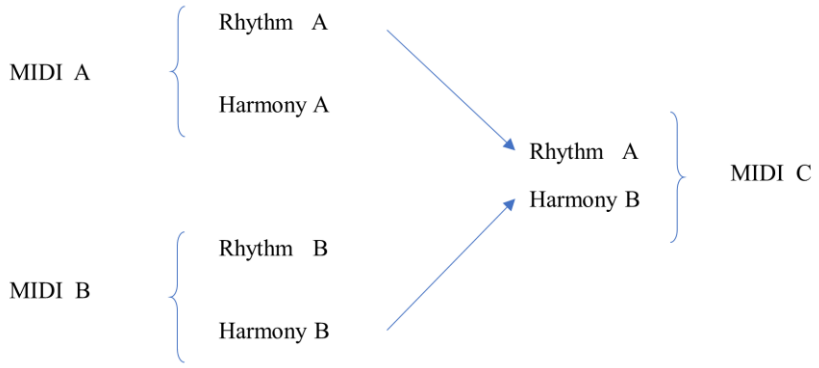


Figure 4-2: Main goal of the designed method. The MIDI file A, with its rhythm and harmony, recombines the rhythmic features with the harmonic feature of a MIDI file B to generate a new MIDI file C with both features of the two files.

The MIDI file has a binary structure that describes *events* that are measured with the unit called *tick* as a discrete representation of the subdivisions of time. The events describe the moment a note is pressed, when it is detained and which pitch was pressed among other descriptors like pressure, attack, etc. all these features are described in time with the *ticks* unit. The main MIDI events used to recombine compositions are the *onset* and *duration* of each note to find the required data. The *pitch* of the note will be represented with the key number corresponding to the MIDI values. These values have a direct relation to frequencies of the sound played on keys in a piano or a keyboard.

The main difference with these notes found in the MIDI file structure is the relation with the *pitch* values of the actual instruments, their primary reference for these pitches is the Piano but with an offset. Middle C *(do)* or C4 corresponding to the key number 40 on the piano of 88 keys and a frequency of 261 Hz has a MIDI note number of 60. The first note on the piano A0 with an approximated frequency of 27.5 Hz has a MIDI note number of 21, and it increases up to a MIDI note value of 108, which corresponds to C8, key number 88 on the piano with an approximated frequency of 4,186 Hz. (Back, 2020) (Suits, 2020) (Stolzenburg, 2015).

The proposed method describes a series of steps to extract features from two compositions and encode them into an abstraction sub-features. Then the next step in the process is to change properties at the encoded space. Finally, by decoding the abstraction, these changes are propagated into the non-encoded space resulting in a new composition with regularity and consistency. This method ensures that a series of notes in a sequence at the original file will have those changes reflected every time the original composition had such a sequence. Those changes will consistently replace notes even at different moments or at different octaves in the harmonic space. These consistent changes generate a composition with a local and global structure. At the moment, a composition is analyzed, and its features are changed, it is called a transformational based compositional method, where a given composition is taken as a seed, and a new composition is obtained by transforming its features (Bigo & Conklin, 2016).

The general method for music data generation uses the MIDI structure both in the analysis of existing data and, as a result, to represent the newly generated data. The process steps for

analysis for feature extraction, the encoding of sub-features, dictionary generations, and recombination are depicted in Figure 4-3.

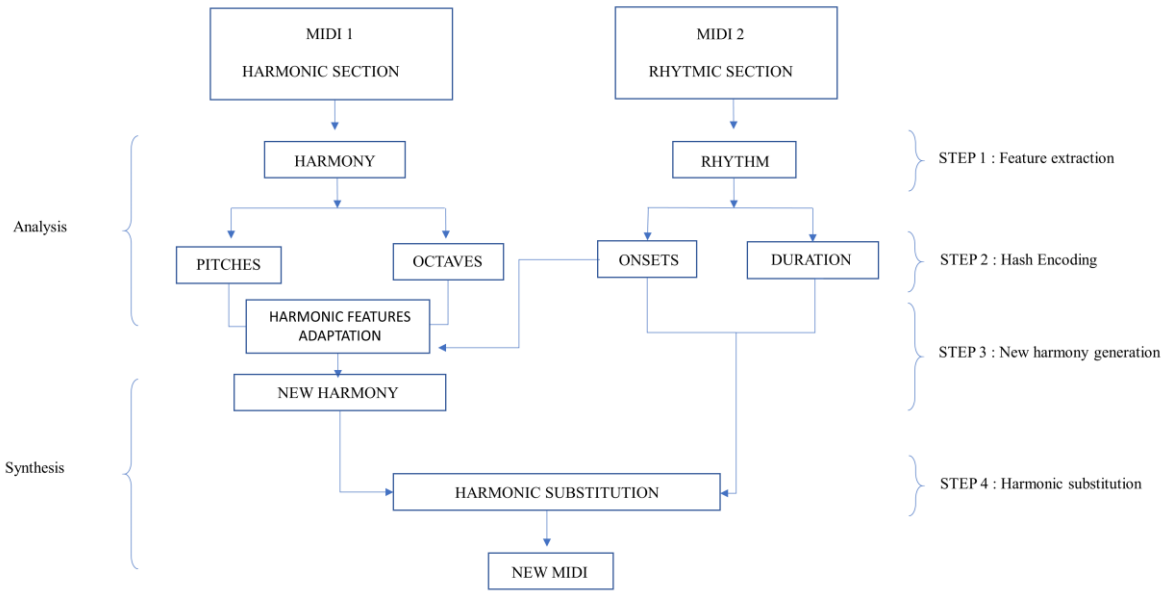


Figure 4-3: The flow chart of the principal steps of the proposed method for dictionary extraction of the music compositions and recombination of the features to create new data

Information found in MIDI events can be listed as *string* structures that represent the instrument name and its numeric value in the MIDI table of instruments. These lists of instruments also contain each of them, at the same time, a list of structures that describe notes with its properties; *Onset*, *Duration*, and *Note Pitch;* see Figure 4-4, and we can visualize the same information as a piano roll representation.

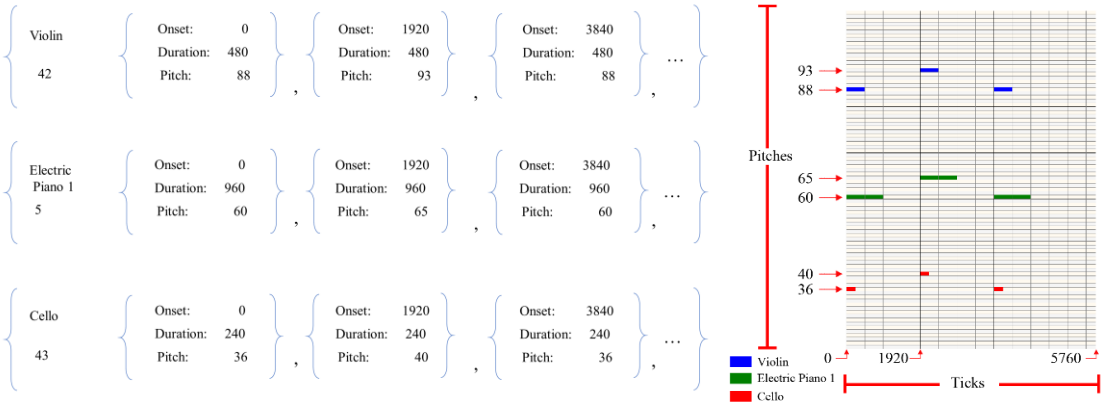


Figure 4-4: (Left) Examples of the information found at the structures of the MIDI events in a composition. This list of instruments contains "Violin", "Electric Piano 1" and "Cello" with corresponding numbers of 42, 5 and 43 and their respective lists of notes played by each instrument. (Right) Examples of the structures of the MIDI events in a composition represented as a piano roll, commonly known as the grid of 128 values on 'y' axis for pitches from 0 to 127 and a horizontal value divided into quarter notes measured in ticks in the 'x' axis. Between tick 0 and the tick 1920, there are four quarter-note spaces of 480 ticks.

When dividing the original MIDI files, every window analyzed has a sequence of notes, this sequence of notes found in each window are called *words*, they must be decoded into their sub-features as depicted in Figure 4-5. A *dictionary* contains all the non-repeated sequences of values (*words*) found in the windows of the divided MIDI file. Every composition contains four different dictionaries corresponding to the four different values of notes: *pitches*, *octaves*, *onsets*, and *durations*. The *alphabet* of the pitches in all sequences, which are the *words* found in every composition, generates the pitch *dictionary* and are the indexes of the pitches values of western music, as depicted in Figure 4-5 and Figure 4-6. (Suits, 2020)

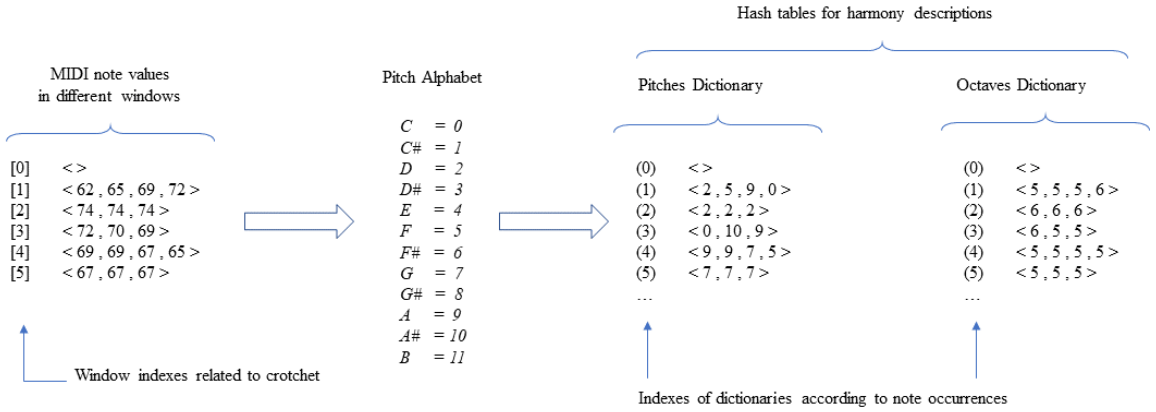


Figure 4-5: MIDI note values found in every indexed window (left) decoded into their respective values of pitch and octave (right) to generate the harmonic dictionaries.

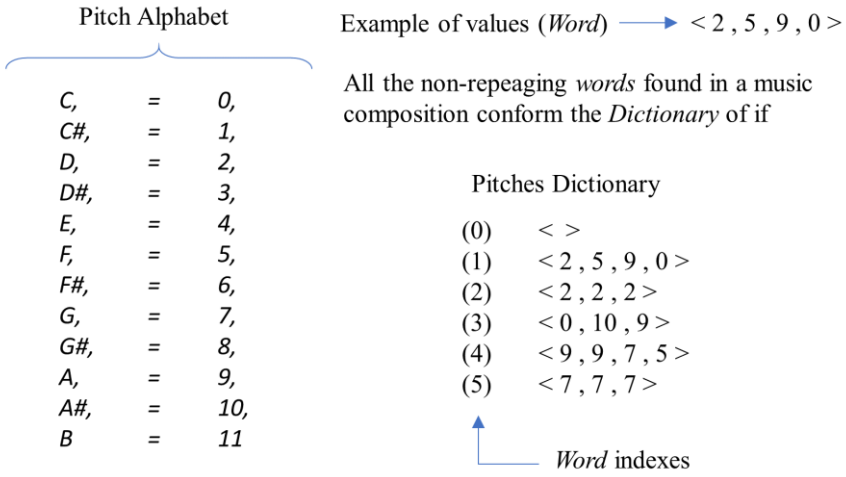


Figure 4-6: Pitch alphabet with the index number of each value (left) and an example of a sequence of values and a dictionary with five different words (right).

A *Map* is a description of the *words* stored in dictionaries and indexes as a sequence of *words*. It describes the composition. An example of *maps* with only one *pitch word*, one-*octave word*, one *onset word*, and one *duration word* is depicted in Figure 4-7.

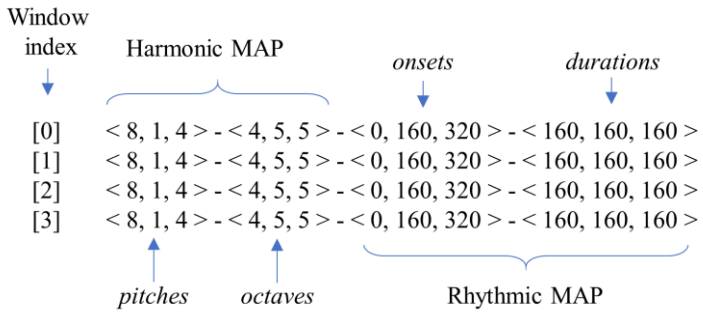


Figure 4-7: Examples of MIDI events decoded into sub-features in a composition represented as a sequence of words in a map.

This proposed method for analysis and extraction of sub-features for encoding is performed on two different songs *C*1 and *C2*. After these extractions, the harmonic development of one composition *C1* described with the *words* found in its *dictionary* is transformed and adjusted to fit into the rhythmic *dictionary* of the other analyzed composition *C2*. Finally, the resulting dictionaries are decoded back, this harmonic recombination of the dictionaries is achieved with genetic algorithms, conveying in a new composition *C3*.

4.3 Steps description for harmonic recombination

The following descriptions cover the model in further detail for each different step of the methods to extract, encode, recombine, and generate a new music data file. Each description contains a diagram and needed definitions and parameters to replicate the complete method of composition.

4.3.1 Feature extraction for harmonic analysis

Feature extraction is the first step of the general process for harmonic recombination, and it is a two-part process. As it was mentioned in Chapter 3, it is about dividing the complete sequence of notes into a list of non-overlapping windows of notes and decode each note into the four sub-features of music from the main features of *harmony* and *rhythm*. The first part starts by dividing the compositions (harmonic and rhythmic) into lists of *words* of notes which are set in every composition. These divisions are excerpts of windows of a size equal to the declared *crotchet* size *q*, which is described in ticks *t* at the header in the file. Each of the excerpts contains notes which are divided into two different types, *harmonic and rhythmic* features. These sub-features are divided again into two descriptors, each of themselves. The *harmonic* feature is divided into *pitches* and *octaves*, and the *rhythmic* feature is divided into *onsets* and *durations*. This encoding is not the best compression rate, because the main goal is not to compress into higher rates the representations but to abstract, the characteristics in different countable features based not on high compression but music features related to the *crotchet*. For example, if we have two consecutive notes, the possible combinations of them are $128\ x\ 128 = 16384$ by diving the features in this subdivisions using as pitches of 12 values and 11 octaves the possible combinations of them are $12\ x\ 12 = 144$ in the first instance and $11\ x\ 11 = 121$ for the second one. We are adding them up to a total of 265. If we keep the MIDI pitch value of 128 values, the possible combinations keep growing by the length of the composition.

A note pitch and octave, which belong to the *harmonic* features, have a standard range of values in any MIDI file. There are 128 possible MIDI pitches in a MIDI file, and the values

are between 0 and 127 each. There are only 12 pitches (C, C#, D, D#, E, F, F#, A, A#, B). There are 11-octave values obtained from those notes, and they are between 0 and 10. The *rhythmic* feature has *duration* and *onset,* and they are related to the declared *crotchet* size q so that it may vary on each file.

Let there be a MIDI file of music composed of a song C of length L in ticks t with a *crotchet* value of q also measured in ticks. After dividing the music composition, the first stage of the encoding, it will have $\left(\frac{L}{q}\right)$ which is the total number of windows of length q each of them. These windows are processed and decoded into the four sub-features described before *pitch, octave, onset,* and *duration.*

Every window w in the composition C can be found with C_w. Each window w is decoded into its sub-features by extracting them from each note inside the sequence found in it. Where $0 \leq w \leq \left(\frac{L}{q}\right)$. Each note has a *pitch* p_{wn}, *octave* c_{wn} an *onset* o_{wn} and, *duration* d_{wn}. Where $0 \leq n \leq s$ and s is the total number of notes in each sequence *(word)* of each window. Each note N has a MIDI value and a starting moment in *ticks* with an offset to declare its *duration*.

To compute the sub-features of the MIDI note N_{wn} we use the Eq. (1). to obtain the *pitch* value, Eq. (2). To obtain the octave, Eq. (3) to obtain relative *onset* to the window and, the duration is equal to the offset δ found in the structure of each processed note as in Eq. (4).

$$pitch_{wn} = N_{wn} \, mod \, 12 \tag{1}$$

$$octave_{wn} = N_{wn} \, / \, 12 \tag{2}$$

$$onset_{wn} = N_{wn}(ticks) \, mod \; q \tag{3}$$

$$duration_{wn} = \delta(N_{wn}) \tag{4}$$

After extracting the sub-features, there is a decoded version of each composition in a list of *windows* with four sequences(*words*) of descriptors. Instead of using a note number (from 0-127) described in a range of *L* moments in ticks from start to end, we have a sequence of descriptors of each note. From the previous example with the *violin*, *electric piano,* and *cello*, we have a *crotchet* size of 480 ticks and their instrument index, where the violin has the 0-index value, the electric piano is 1-index value, and the cello has a 2-index value. Then, there is a list of MIDI notes in each of the instruments. It becomes the list of *words* after dividing the list into every 480 ticks, as depicted in Figure 4-8.

Instrument	Word 1	Word 2	Word 3
0	{ 0, 480, 88 }	{ 1920, 480, 93 }	{ 3840, 480, 88}
1	{ 0, 960, 60 }	{ 1920, 960, 65 }	{ 3840, 960, 60}
2	{ 0, 240, 36 }	{ 1920, 240, 40 }	{ 3840, 240, 36}

Figure 4-8: Examples of the structures of the MIDI events found in a composition represented in a list of sequence numbers that describe notes each found in a quarter note of 480 ticks, we can observe the instrument index on the left and the sequence of notes found in each of them.

If we take the values of the pitches of the first *words* found in the first 480 ticks, they are transformed as follows; the first two numbers remain the same because they belong to the rhythmic description, but the 88, 60, and 36 MIDI values are transformed. The value 88 is transformed into 4 using Eq. 1, and then by using Eq. 2, it is obtained the number 7, the result of every first note of the three instruments is depicted in Figure 4-9.

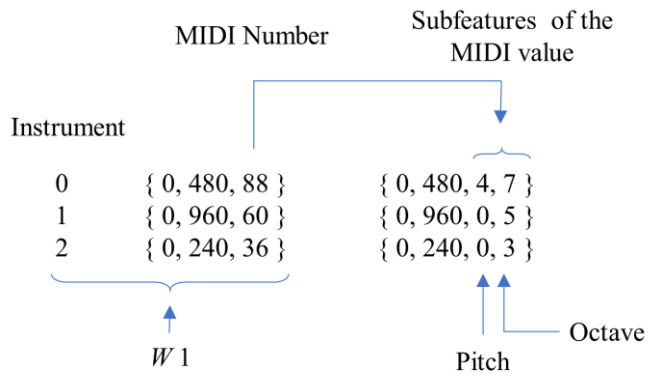


Figure 4-9: Examples of the three first structures of the MIDI events found at the first window (w1) in a composition represented as note descriptors and equivalences of the harmonic values of the MIDI number transformed into pitch and octave.

The main advantage of this *feature extraction* method is that it is dependent on the quarter note resolution of the file. Thus it does not need to encode several times the file to select from the different obtained representations the higher rate as in the COSIATEC algorithm (Meredith, 2014).

A possible disadvantage in polyphonic music analysis is that the files could have specific MIDI messages that could change the *percussion* channel, this could provide an incorrect abstraction of the features given that the *percussion* can be described in the harmonic channels. A previous selection or process of the files would have to be done to avoid these kinds of problems when selecting files from the internet since they are not standardized for this kind of situation. If this is detected, a preprocessing of the file is necessary, arranging the *percussion* into the 10^{th} channel.

4.3.2 Encoding steps for harmonic recombination

The *encoding* is done after diving the notes in windows and extracting the sub-features from the analyzed compositions. This process abstracts lists of descriptors from the excerpts into non-repeating sequences and groups them according to the sub-feature in corresponding *dictionaries*. Finally, it *encodes* the map of the composition by replacing each sequence with the corresponding index of their corresponding *dictionary*.

This encoding reduces the number of different sequences of sub-features of notes called *words* into only those that are used to describe the composition. Each of the sub-features generates a different *dictionary* with the different *words* found in the composition. These *words* are in every window of all the instruments in the music file. After finishing the encoding step, there are four different dictionaries, each for every different descriptor of *rhythm* and *harmony* and a new encoded *map*, as depicted in Figure 4-10.

An advantage in polyphonic music is that by encoding with this strategy, it does not matter how many instruments there are in the music piece, all the windows are encoded into only one dictionary of each (*pitch*, *octave*, *onset*, and *duration*).

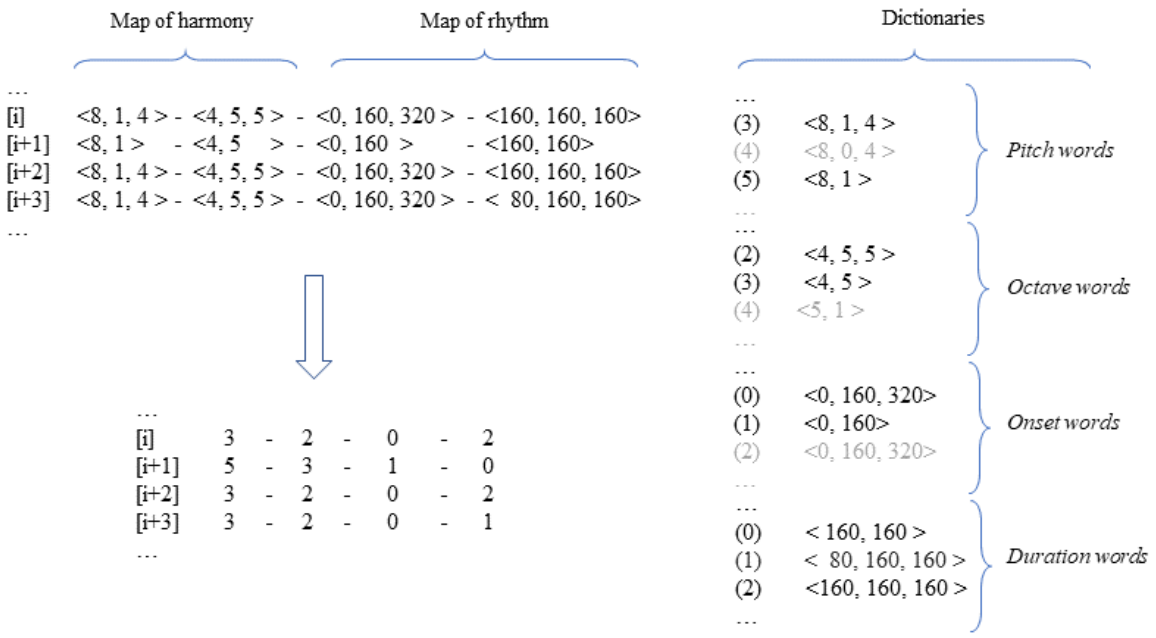


Figure 4-10: Example of a map from an instrument (left) and the four dictionaries with the words of the sub-features which contains the words used in the composition to describe it. The map is encoded with the indexes on the dictionaries that represent each word.

The possible problem is that the divisions of the main channels of each instrument should not be mixed when analyzed, as in the case of the *percussion* section might get wrongly encoded and mixed along with the harmonic development. It is necessary to select the files correctly to be used. If channels are mixed, then it will have to be extracted and separated the channels, in a preprocess, to avoid mixing harmonic and rhythmic sections. If it is something that the user needs to preserve, then the rhythmic section would have to be correctly assigned to the *percussive* channel.

4.3.3 Generation of a new harmonic dictionary for recombination

The third step in the process of automatic music composition is the generation of a new *harmonic dictionary* based on an existing music file to fit it into a given *rhythmic dictionary* of a different composition. Since the *harmonic dictionary* is a list of non-repeating combinations of *words(sequences)* found in a composition, it is different from the *harmonic dictionary* of other compositions. Inconsistencies between different dictionaries often present problems of adaptation in the size of the dictionary and the length of each *word* in it.

Dictionaries of different songs contain not only different *words* but different lengths in each of them. Each dictionary varies in the number of *words* found in each *dictionary* and the number of notes in each word (see Figure 4-11). Let us call the number of words in a dictionary, the *global features*, and the length of each sequence the *local features* of the dictionary.

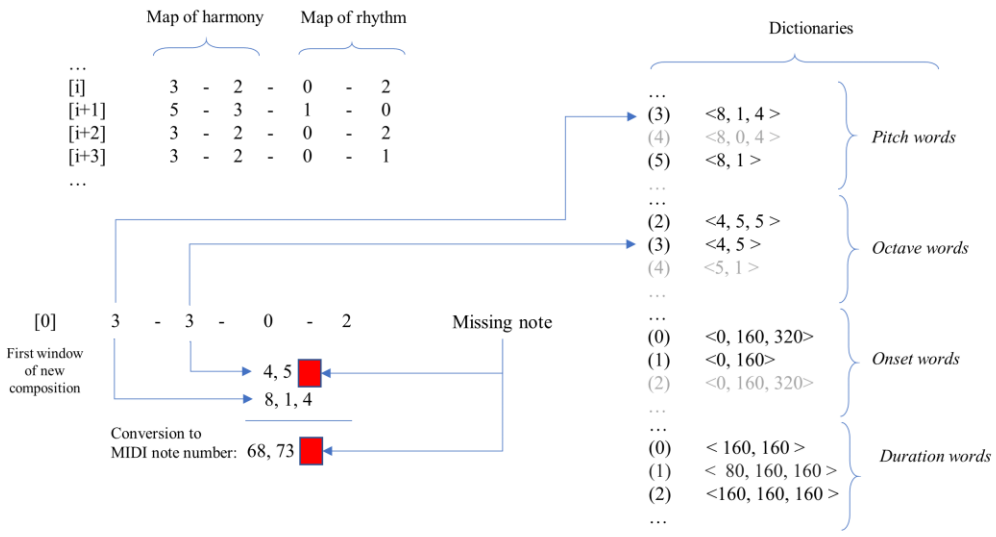


Figure 4-11: Example of inconsistency in the length of sequences at reconstruction

The first attempt for adjustment is a random repetition or deletion of notes in a *word* to fit the length of a second word, and it can generate inconsistencies in the harmonic development of the resulting composition. The *words* in a dictionary are used several times along the composition. If every time we need to adjust a *word,* random action is taken, then the global consistency is lost because every time may be different. A random decision is going to vary the result, as is shown in Figure 4-12.

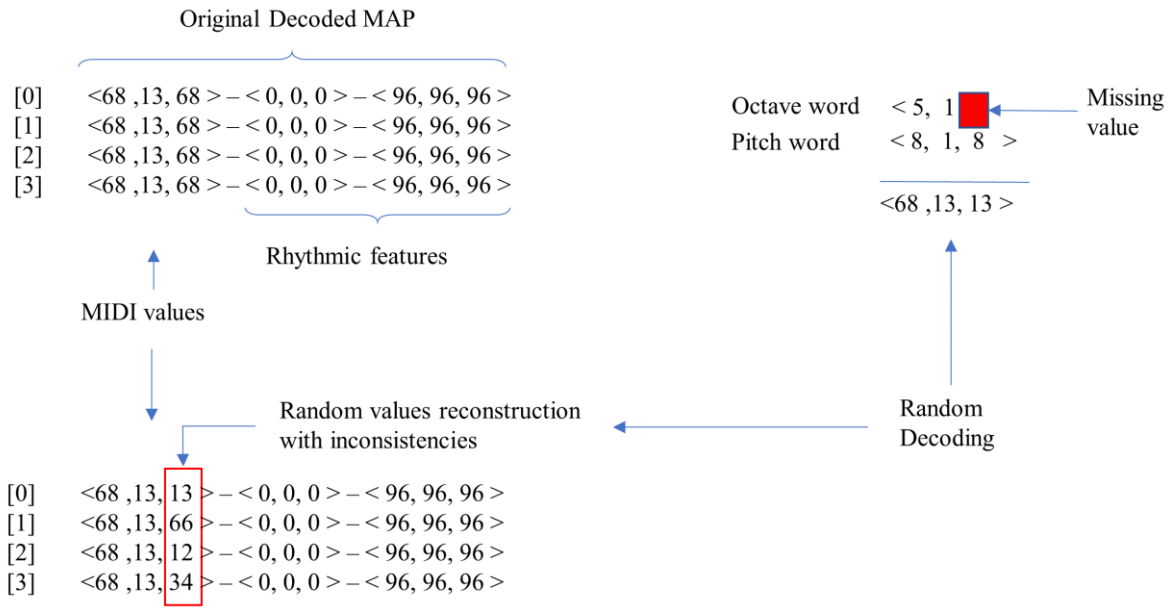


Figure 4-12: Example of two words from the harmonic feature with no compatibility decoding with random values. Decoding is not consistent due to random selections.

The generation of a new *harmonic* dictionary instead of a new *rhythmic* dictionary is due to the possible values in every MIDI file that are the same, unlike the *rhythmic* values which might have a different resolution in ticks with different individual values in every composition. A genetic algorithm is used to generate dictionaries with *global* and *local*

constraints. Using *global* and *local* metrics, it solves the adaptation problem by constraining each individual of the population in its size and the length of sequences in it to fit into a specific rhythmic dictionary.

The method starts with two compositions $C1$ and $C2$; they each have their dictionaries, the *rhythmic section*, and the *harmonic section*. Let us call them $r1, h1$ and $r2, h2$, respectively. The objective is to generate a new composition $C3$. To achieve the generation, we need to fit *h1* with *r2*. To achieve this, global and local features of the generated dictionary would have to be the same as the local and global features of *h2*. The genetic algorithm aims to find a new set of *words*, x, for each in the target *dictionary* that minimizes the fitness function and satisfies the constraints.

The objective function of the optimization problem is to find a solution x that minimizes a vector Y that represents four histograms of the *seed* dictionary *h1*, as closely as possible. Those histograms of the individuals of the population (*h3* candidates) are compared to the *harmonic dictionaries h1* of the selected composition $C1$.

The *dictionary* used as the *seed* to start the genetic algorithm is analyzed at two different resolutions. First, a global histogram of the values in the dictionary is computed if the target is the *pitch* dictionary. It generates a 12 continuous value array as the global histogram, used for the global constrain of 12 possible pitches. The vector Y contains 48 values from four histograms; one histogram is computed by counting the occurrences of all the pitches in all the pitch dictionary as a *global histogram* with 12 continuous values using Eq. 5, where p corresponds to the index in the histogram equal to the pitch value.

$$H[p] = \frac{number\ of\ values\ of\ pitch\ p}{total\ number\ of\ values}, \qquad (p = 0,1,\ldots 11) \tag{5}$$

The dictionary is also divided into three segments, and it is generated a histogram for each segment. Each histogram represents a countable behavior of the pitches in that particular segment of the dictionary as a local histogram. If the target is the *octave* dictionary. It generates an *eleven* continuous value array for each histogram.

The generation starts by analyzing the *harmonic* dictionary *h2* to be replaced, which is the *dictionary* with the *global* and *local* target *constraints*. The constraints to generate new candidates are the *local* features (number of notes in every sequence in the dictionary) and the *global* feature (number of sequences in the dictionary).

The vector Y that represents the four histograms is the measurable feature of each generated *candidate,* and it is used in the fitness function measured with a mean square error (*MSE*) metric described in Eq. 6 (Li & Zhao, 2001b).

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(Y_k - \widehat{Y_k})^2 \tag{6}$$

Where Y is the obtained histogram of each candidate in the population, $\hat{Y}$ has the target histograms of the dictionary used as seed, and k is the length of the vector.

The generation of the *candidates* is done by using the probability density function from the original *seed* dictionary *h1*, by randomly picking elements from it with constant random. This generation creates a population where every candidate meets the required constraints, and their corresponding histograms are close to the target.

The same process is repeated to create the *octave* dictionary, providing variation in every generated composition. With only one generation of 1000 candidates suffice to select a suitable candidate to fit the *rhythmic dictionary*.

Further generations can be recombined or generated, but the result presents no perceptual difference. The speed of the algorithm is due to the use of the probability density function to generate each individual. There is no crossing function and no mutation parameters. Using this technique, every run of the method generates variations of the dictionaries resulting in different compositions. If the same composition is used in analysis and as seed for the genetic algorithm, the result is a variation of the original file. The variation is reflected in the harmonic progression by changing the order of the pitches, the octaves, or both.

Since it is not using several generations (which can be used, it is not a restriction), the actual generation and tryouts for different solutions should not be expensive. Also, the recommendation is to use the probability density function in the *seed* dictionary to generate a more adjusted dictionary. When the selected dictionary is set, it could have a better fit to be recombined. This method can be challenging to implement due to the indexes. It needs to correctly generate each dictionary by each individual along with the imposed constraints of the generation. It is recommended to try several times to select a solution suitable for the desired outcome, and this is the most time-consuming step.

4.3.4 Harmonic word replacement

The last step is the *Harmonic word replacement*, which is done by replacing the *harmonic* dictionaries, both *octaves*, and *pitches*, in the file used for their rhythmic base. It is the same

as decoding the dictionaries back to the original music composition. The pitches are replaced with the new ones of the new harmonic development from the MIDI file used as seed. An example of the result is depicted in Figure 4-13.

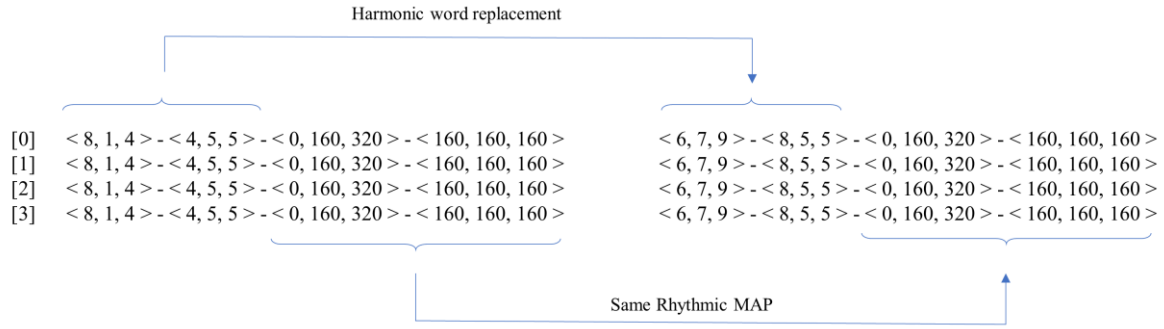


Figure 4-13: Word replacement in rhythmic map.

In chapter 6, results are presented to demonstrate the consistency of the replacement of the notes given a *seed* dictionary and establishing a *rhythmic dictionary* as a first MIDI file. The sequences show different results depending on the replaced dictionaries.

Even with careful selection of the files, if they have a MIDI message not recognizable to the developed software, it could cause undesired outcomes. All this process is susceptible to the selected files. This method could be translated to other music descriptors like music XML, for example, as long as it is described the *crochet* resolution of the messages, and there is a similar representation in ticks of time and the note pitch.

This model is one of the first methodologies capable of combining two different music composition files to generate a new composition. Previous research like (Herremans & Chew, 2019) proposed a methodology to generate variations of a piece, also taking a composition as a *seed*. The main advantage of the proposed method is that unlike the tension profile, it

can generate new pieces directly from different musical compositions. The constraint that a variation method has is due to the sizes a file has, making it not usable for other harmonic development. Another advantage is the dictionary abstraction, the method used for the abstraction of the notes.

It keeps the rhythmic development with a method designed to find patterns along with the composition (Meredith, 2014). Still, the reference is the compression rate, which makes it is more time-consuming. It is more time-consuming because they create different compressions, and then the one with higher compression is selected. In the proposed method, on the other hand, we are also able to keep the rhythmic development and the melodic or harmonic patterns but based on the self-description of the music file. The experimental results with the rhythmic shape kept, are shown along with different changes expected according to the different number of dictionaries replaced.

The method can be *tested* with two metrics one for the harmonic and another one for the rhythmic features. The harmonic value is set by the threshold used in the genetic algorithm already. The *rhythm feature* still needs to be tested at the end of the generation to validate the method. In the original file, *C2* selected to use its *rhythm;* the *onset* and *duration* features have to be the same in *C3* and evaluated with Eq.(7). This property is shown in the results chapter 6 of experiments in music synthesis.

$$\sum_{k=1}^{n}(ro2_k - ro3_k) + \sum_{k=1}^{n}(rd2_k - rd3_k) = 0 \qquad (7)$$

Where *ro2* corresponds to the rhythmic subfeatures of onsets of *C2*, and *ro3* corresponds to the same subfeature of durations in *C3*.

4.4 Discussion

The method provides a complete process of harmonic replacement with a genetic algorithm. It has a fitness function, which is a metric in itself and also a rhythmic equation to verify the correctness of the rhythmic coupling. Still, these metrics present a range of correctness of the process, but the final selection of the requirements or the best fit of harmonic and rhythmic substitution will be decided by the final user. This selection is the main limitation of the process because there are no metrics for music perception. The process is bound to the provided files and the ability of the user to select music compositions with desired features.

Chapter 5 Results of tests of the proposed approach for feature analysis

5.1 Collision detection and random initialization

In this set of experiments, we analyzed the detection of vector collisions to visualize how many vectors are grouped. It is an essential step of our proposal because sometimes multiple vectors cannot be seen if they are plotted to the same point in space as well as the clusters are not comparable if they contain one or multiple vectors. This collision detection helps to know whether harmonies are representative in the composition, or they are an outlier when the density of vectors in a cluster is low. Notably, the normalized distances with an initial state of random positions have been compared with initial positions given by 12 points of an icosphere, which correspond to the 12 notes. The convergence of the projection got faster with the initial state of random positions.

The projection reached a balanced state in which vectors start a loop. After reaching convergence, there were some vectors changing clusters, but the number of clusters remained the same (see Figure 5-1). Before initializing the iterations process and after computing the corresponding to music piece SSM, we tested both starting points for the vectors. In our approach, random and fixed initializations have been tested. One starting point was considered as a random position for each of the vectors and another one by setting a fixed position to each vector given by its assigned 3D point of hue value depicted in Figure 3-6.

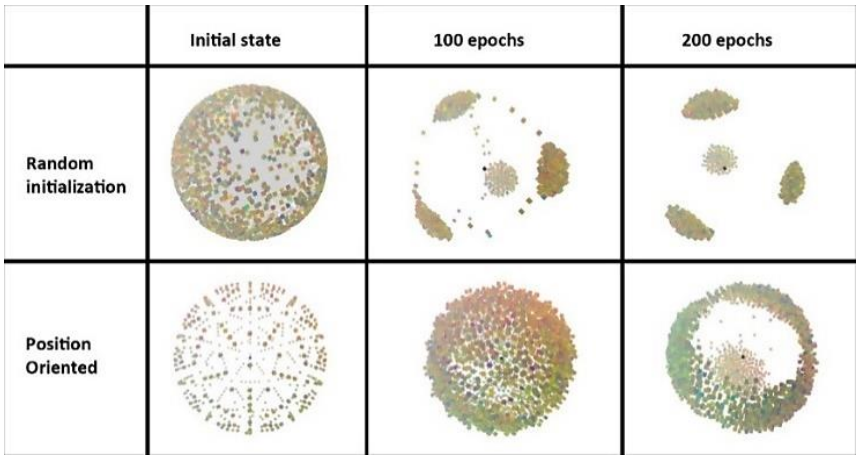


Figure 5-1: Hue colored 12-dimensional vectors of all the possible 4096 combinations. Initial state (left), 100 epochs (middle), and 200 epochs (right) for random and position-oriented initialization.

The obtained results show that with every random initialization, the projection converges faster than with the position-oriented one. With position-oriented initialization at 200 epochs, the vectors just started to set the clusters, and with the random initialization, this was achieved at 100 epochs. By the iteration 200, the random initialization reached a stable projection with four clusters, and the fixed initialization of points was still ordering the vectors.

A space of 3-orthogonal planes of size *s* is used to visualize each of the tonal 3D vectors. Mainly, 3D projection can be used in virtual environments like videogames with associated graphic libraries. The virtual camera is not of fixed-view, and it could be adjusted to positions for the best visualization of the results in different camera angles. It is done by dragging a mouse over a drawing canvas to rotate the resulting projected vectors. Thus, the proposed tool provides multisided visualization instead of a static representation of data vectors. In the presented Figure 5-1 as well in the next figures, the camera was adjusted to visualize results by showing the most significant number of vectors, fixing the silent vector at the center with

black color. That provides better visualization of stems concerning themselves and the silent vector.

The following minimum criterion must be used for collision detection in three-dimensional space, particularly, the distance between two vectors cannot be smaller than *s*. If the distance is less than *s/2*, both vectors are separated by $\alpha = s/2$ using (4). For example, if there are two vectors in 12 dimensions described as *v1=[1,0,1,0,1,0,0,1,0,0,0,0]* and *v2=[0,0,1,0,1,0,0,0,0,1,0,0]* with an *Euclidean* distance *d1=1.732* between them so, there are corresponding vectors in 3D space with random initialization, which are depicted as $v1_{3D}$*=[0.3, 0.5, 0.3]* and $v2_{3D}$*=[0.3, 0.7, 0.2]* with a corresponding *Euclidean* distance of *d2=0.223*.

If size *s* of each 3-sided plane is equal to *s=3*, the value of *d2* is smaller than the size of s/2 so, the distance between two 3D vectors is set by the linear function shown in (4). Since *s=3* then $\alpha = 1.5$ and by applying (Eq. 4 from Chapter 3) we get the new values for $v1_{3D}$*= [0.3, 0.4, 0.35]* and $v2_{3D}$*= [0.3, 0.85, 0.125]* with a new distance *d2= 0.503.* As a result, it is still smaller than the size of s/2, which is the size of 3 orthogonal planes.

This process is applied only once because of the next iteration. When it is re-evaluated, both vectors will be updated again, generating values *v3= [0.3, 0.175, 0.4625]* and *v4= [0.3, 1.187, -0.04375]* with *d2=1.132*, which is still smaller than *s/2* and *d1*. At the following iteration both vectors are updated, and their new values will be *v3= [0.3, -0.331, 0.715]* and *v4= [0.3, 1.95, -0.423]* with *d2=2.55,* which is finally bigger than *d1* and s/2.

Some famous musical pieces have been processed. First, the method splits the composition to evaluate the proposed method for music visualization based on the spherical projection

into each music *bar/measure* found in the file. Then each measure is subdivided in a non-overlapping window with the size of the crotchet declared in the MIDI file. In Figure 5-2, we can observe projections of piano sonatas by different classical composers using Euclidean distance to create the corresponding SSM projection. Similarly, Figure 5-3 shows some compositions of famous movie soundtracks applying our spherical projection method.

At first sight, we can observe the higher complexity in the use of different harmonies expressed by vector sparsity or density at 3D projection. The colors represent tones, and similar colors have similar tones. For example, in both Mozart and Chopin sonatas, the density is visually higher than the density in *Moonlight Sonata* of Beethoven. This difference is because, in Moonlight Sonata, the same combinations of notes are used through the entire music composition.

We can observe a single yellow vector at the bottom of the Beethoven projection in Figure 5-2, which represents a *B* note colored according to the rules presented in Figure 3-7. This *B* note is not repeated and is not shared with other notes; otherwise, several vectors would be closer to it, but it is isolated. In the projection of Yann Tiersen composition in Figure 5-3 we can observe even lower vector density comparing it to the density of "*Moonlight Sonata"* from Figure 5-2

5.2 Proposed Visualization vs. Autoencoder

The autoencoder is selected for the experiment to compare the efficiency of the two proposed methods of dimensionality reduction and visualization. It is the best frequently referenced method. An autoencoder has been used to process 4096 combinations for 12 notes represented by vectors in the 12-dimensional space.

An autoencoder consists of an input layer followed by a layer of three neurons as a bottleneck *(x, y, z)*. The third layer on its input receives data representing three-dimension vectors, and the last layer is used for the generation of outputs, which correspond to the representation of vectors in 12 dimensions.

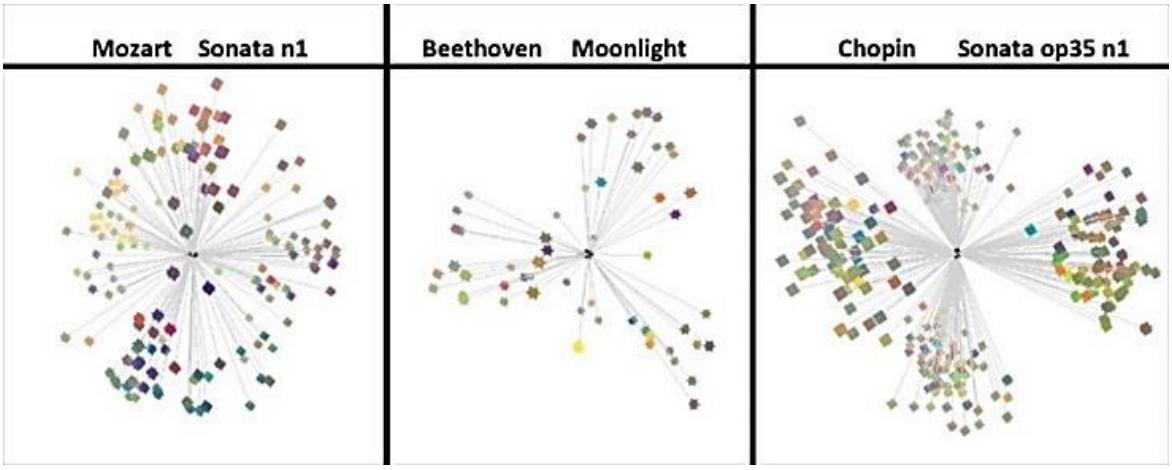


Figure 5-2: A Piano sonata by Mozart, Beethoven, and Chopin, respectively.

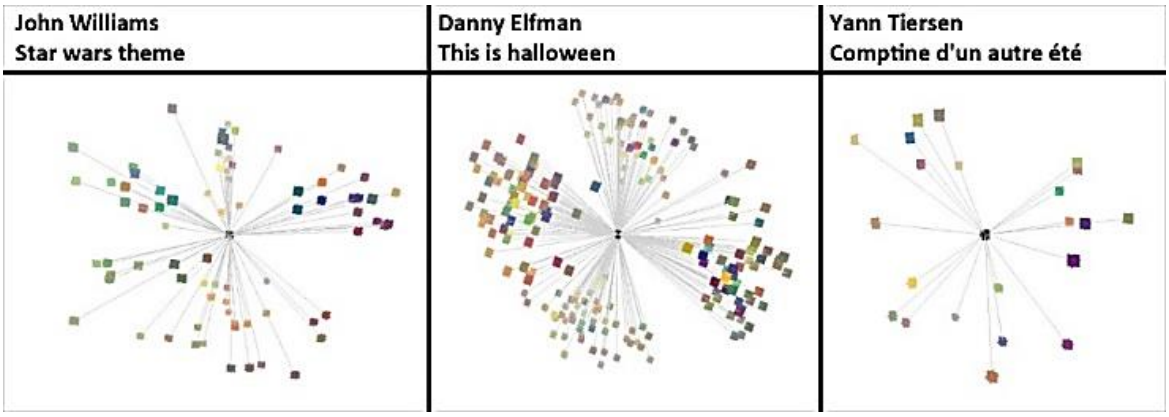


Figure 5-3: A Soundtrack composition by John Williams, Danny Elfman, and Yann Tiersen, respectively.

An autoencoder uses MSE with a learning rate set to 0.01, which empirically is not too high to keep jumping or too low to stuck in local minima (Theis et al., 2017). The loss of the autoencoder keeps decreasing, but the adjustment of deltas stops around the 31,000 epochs. The autoencoder splits the projected vectors shaping a cube (see Figure 5-4), which has no relationship with a metric perception of harmonies in music composition, and the visualization reference has no related appearance with harmonic features.

MSE and AAE errors are computed according to (6) and (7), respectively. Figure 5-4 depicts the results of the iteration process by the autoencoder, and it can be visually compared to the results of the proposed approach shown in Figure 5-1.

The autoencoder reduces dimensionality faster with a smaller error. Still, it cannot be tuned to measure harmonic relations with mathematical representations. Our method works with SSM of vectors computed without restrictions to specific metrics. The SSM works as a bridge for vectors visualization from higher-dimensional space into lower-dimensional space. It is computed to test metrics and observe their projected behavior.

The epochs are the number of cycles that each of the adjustment methods employs. Every time all the tonal vectors are adjusted, the error fitness process is updated, and an epoch is counted. Although the autoencoder has a lower error in the fitness function, it is not adjustable as our approach, which can change the distance function to generate new SSM and produce new projection that can fit the desire relationships. At the projection of the 4096 tonal combinations in 12 dimensions, both errors generated from the autoencoder were MSE = 3.18% and AAE = 0.005% with 31,000 epochs.

Our method had only 300 epochs to get AAE = 0.67 set by hand as a threshold with an MSE ≈ 10%. The MSE presented in our method is directly related to the selected distance, unlike the MSE presented in the autoencoder, which is the error obtained by the nonlinear compression by the layers in the neural network. This MSE approximates the error of the projection of the fully connected layers. Still, they are not directly related to the higher dimensionality reduction and music visualization as our proposed method. The absence of correlation of the errors does not allow a direct quantitative comparison

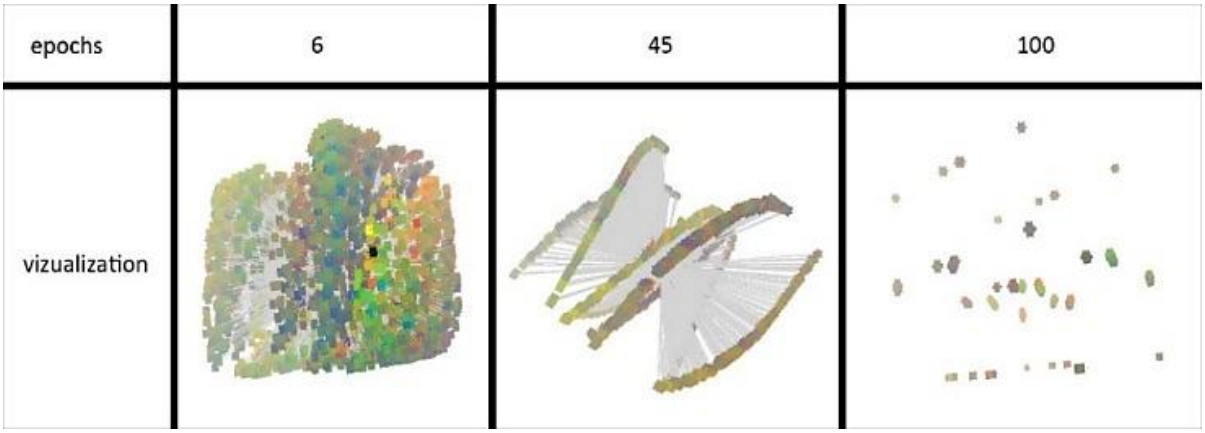


Figure 5-4: Visualization of the projected vectors by autoencoder at different iterations of the process of reducing the 4096 harmonies from the higher dimension.

On the other hand, the clusters generated by the autoencoder could have a problematic interpretation during visualization. It is essential to mention that the proposed method has an additional advantage, which consists of a straightforward process of selection and adjustment of metrics used for the analysis of the behavior of dimensionality reduction.

In the next experiment, we have selected Hamming distance metrics to compare it with the previously used Euclidean distance. Without any changes in the main steps of the proposed method depicted in Figure 3-1, a new self-similarity matrix corresponding to Hamming distance is computed. The obtained results applying Hamming distance metrics are shown in Figure 5-5, comparing them to results obtained by exploiting Euclidean distance. The obtained results show that both projections are quite similar and look like scaled versions of each other (see Figure 5-5).

Since colors represent notes, in our approach, similar colors represent similar sounds (because the sounds are created by mixing notes), which are clustered in the projections. In Figure 5-6, we can observe combinations of the vectors with different density. In a jazz composition called *Blue train* by John Coltrane, a sparsity of vectors is regular, while a mixture with fewer combinations is observed in the song "*Whiter shade of pale"* by Procol Harum. This sparsity difference means that Blue train uses more harmonic combinations of twelve notes through the composition than "*Whiter shade of pale."*

The proposed method considers silence and keeps it at the center with a proportional distance to each combination of the notes. This use of silence means it preserves distances between harmonies related between them and to silence. That is not considered in well-known scientific reports.

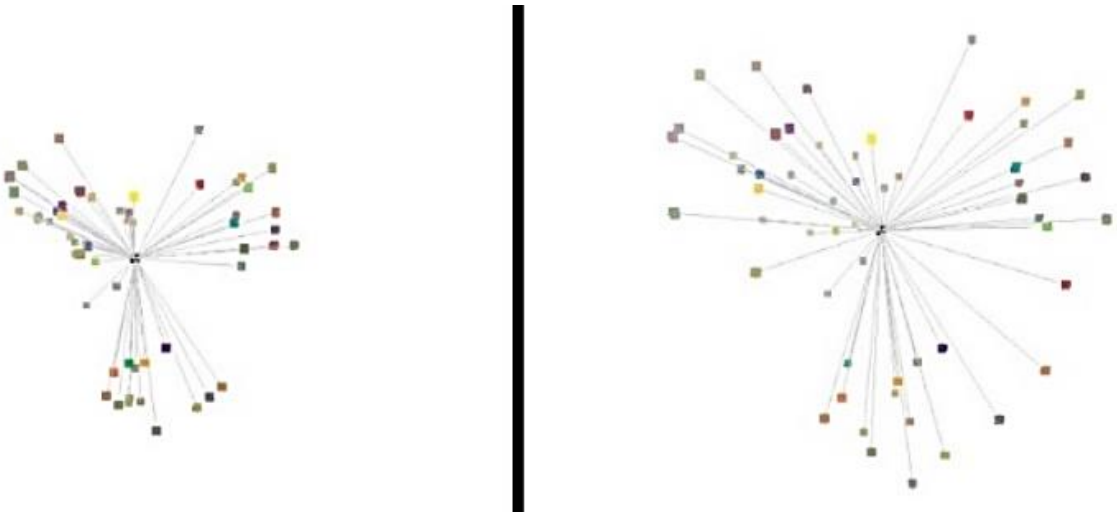


Figure 5-5: Spherical projection of Moonlight Sonata by Beethoven: Euclidean distance took 103 iterations (left), and Hamming distance took 201 iterations (right).

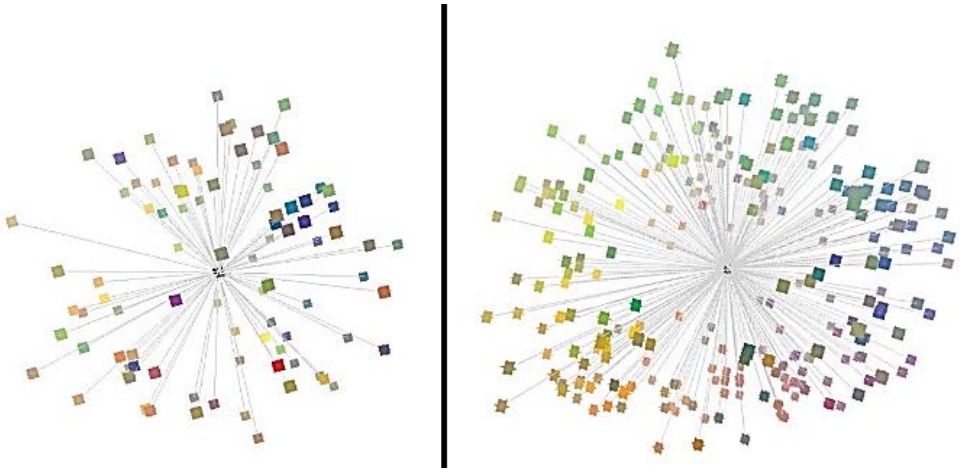


Figure 5-6: Spherical projection of compositions: Whiter shade of pale by Procol Harum (left) and Blue train by John Coltrane (right).

The precision is defined by the MSE and AAE loss during the projection from the original dimension to the 3D space. We could manually change the metrics used to generate the SSM, and all the procedures of the methods for dimensionality reduction will remain the same.

That provides flexibility to explore different relations of harmonics representations in the music compositions.

5.3 Discussion

Although these methods were developed to analyze, extract data features and dimensionality reduction, we obtained a visualization tool when reducing the dimensions into 3, 2, or 1 dimension to decide the use of a metric or to look for a useful behavior in the selected data. These methods are interactively robust as an intermediate step into the primary goal of creating new data by recombining what is found as essential features. The proposed methods determine different kinds of features used for visualizations to choose one that better fit to explore a specific harmonic problem. Mainly, it has been exploited for the tonal representation of music compositions in 3D plots. A self-similarity matrix SSM based on the Euclidian and Hamming distances between vectors in 12-dimensional space has been proposed to use. Smooth and decreasing-only behavior of adjustment average error AAE, which relates dimensionality reduction errors with iterations during setting vectors in 3D space, has been adopted as a stop criterion of the algorithm for adjustment of vector positions. It facilitates finding a threshold for the projection of reduced spatial dimensionality with an error of less than 10% after only 20 iterating cycles.

The Hamming distance is quite simple to use for practical implementation due to its binary representation. In contrast, for visualization, the Euclidean distance provides a scaled result proportional to the square root. Both provide a similar spatial but scaled vector magnitude

projections. The main difference is the more straightforward implementation of the Hamming distance and, as a consequence, the higher speed of dimensionality reduction.

It was discovered that the autoencoder is faster because it needs fewer epochs to converge. Still, it reduces dimensionality redundancy using entirely numerical analysis based on nonlinear relation inside the neural network. It also has a stop criterion, but the reduction is not based on specific metrics designed for data analysis or for interpreting the music data. The proposed method can make low dimensional projection and expose visual patterns applying the same process for operating with different parameters used for performance analysis. Additionally, it considers the silence as a vector, and all the relations of the vectors are set around it. No other methods have been found that consider silence; however, it is an integral part of the music composition process in general. Mainly, silence is considered as the current starting point for a music piece and must be counted before sound begins.

The theoretical contributions consist of using the crotchet declared in the MIDI files as a self-reference to set a size of a time window to split music compositions. Then the reduction of all possible combinations of harmonies in 12D space is applied to convert them into a finite set of 3D vectors, which can be used for analysis and visualization of harmonic features in a file. The practical contributions include an original technique developed for the projection of music composition features in 3-dimensional space based on a self-similarity matrix of high dimensional vectors.

In appendix A and B, further examples of spherical projection and feature extraction are shown.

Chapter 6 Music data generation experiments and results

6.1 Experimental results and evaluation of proposed harmonic recombination method

These experiments were programmed in the .net framework with the C# language. Running on a Windows 10 system of 64 bits, with different MIDI files from the database of classical music (Kunstderfuge, 2019). The machine has an Intel(x) Xeon® CPU E5-2603 v3 @ 1.6GHz. With 24 GB of RAM.

The metric to use in the genetic algorithm to approximate to the harmonic development is the mean square error from each generated histogram of the individuals of the population to the desired dictionary of the harmonic *seed*. Since the generation is constrained to the word length of every element in the rhythmic dictionary, the expected result is a dictionary that fits the *words* of the original harmonic dictionary.

If the new *harmonic* dictionary has the correct word size and length of the dictionary (*local constraints and global constraint*), this means that the rhythmic development was kept. The new harmony fits the rhythm, and finally, if this is accomplished, by decompressing back the file, we have a new file.

In the following example, a pitches dictionary *Dp1* with a length of 21 *words* from a composition *C1* is depicted in Figure 6-1. It is used in a step by step example on how to analyze its *global* and *local* features to use them in the fitness function to approach a structure as the second pitches *dictionary Dp2*. *Dp1* and *Dp2* are used to generate *Dp3*, a dictionary with the *local* and *global* constraints of *Dp2* and with the *local* and *global* features of *Dp1*.

Pitches Dictionary 1

(0) < 7 >	(8) < 5 >	(15) < 9 , 10 >
(2) < 2 >	(9) < 9 >	(16) < 10 , 9 >
(3) < 4 >	(10) < 10 >	(17) < 7 , 9 >
(4) < 0 >	(11) < 3 >	(18) < 0 , 10 >
(5) < 5 , 4 >	(12) < 4 , 2 >	(19) < 10 , 9 , 7 >
(6) < 2 , 0 >	(13) < 6 >	(20) < 2 , 4 >
(7) < 11 >	(14) < 5 , 7 >	(21) < 9 , 7 >

Pitches Dictionary 2

(0) < 2 , 9 , 5 >	(12) < 5 >	(24) < 2 , 9 , 2 >
(1) < >	(13) < 0 , 7 , 4 >	(25) < 0 , 7 , 0 >
(2) < 5 , 2 , 9 >	(14) < 0 , 9 , 5 , 5 , 0 , 9 >	(26) < 5 , 0 , 5 >
(3) < 0 , 7 >	(15) < 7 , 5 >	(27) < 10 , 5 , 10 >
(4) < 0 , 9 >	(16) < 9 , 5 , 4 , 9 , 5 , 2 >	(28) < 7 , 0 >
(5) < 9 , 5 >	(17) < 2 , 5 , 10 >	(29) < 4 , 4 >
(6) < 5 , 9 , 2 >	(18) < 0 >	(30) < 9 , 4 , 9 >
(7) < 4 >	(19) < 7 , 4 , 0 >	(31) < 4 , 2 >
(8) < 2 >	(20) < 9 , 0 , 5 >	(32) < 9 >
(9) < 9 , 2 , 5 >	(21) < 7 >	(33) < 4 , 7 >
(10) < 2 , 5 >	(22) < 9 , 5 , 4 >	(34) < 5 , 9 >
(11) < 0 , 7 , 4 , 4 , 0 , 7 >	(23) < 9 , 5 , 2 >	(35) < 2 , 4 >

Figure 6-1: Pitches dictionary 1 (*left)* of *C1* with 21 *words* and the Pitches dictionary 2 *(right)* of *C2*, which has the structure to fit dictionary 1 behavior.

Every index in *Dp1* returns the corresponding *word w* from the dictionary. The *total number of pitches tp* is obtained by adding up the lengths of every *word* in *Dp1* with Eq. (1). Where *n* is the total amount of *words* in the *Dp1*, and *k* is the index iterating in the dictionary.

$$tp = \sum_{k=1}^{n} length(Dp1_k) \tag{1}$$

Then the pitch occurrence is added up and stored in a vector where the index corresponds to the value of the pitch, as depicted in Figure 6-2. To obtain the vector *H1* with the following values from the example $H1 = [3, 0, 4, 1, 4, 3, 1, 5, 0, 6, 5, 1]$ these values are obtained by using Eq. 2.

$$H[p] = \frac{number\ of\ values\ of\ pitch\ p}{total\ number\ of\ values}, \qquad (p = 0,1,\dots 11) \tag{2}$$

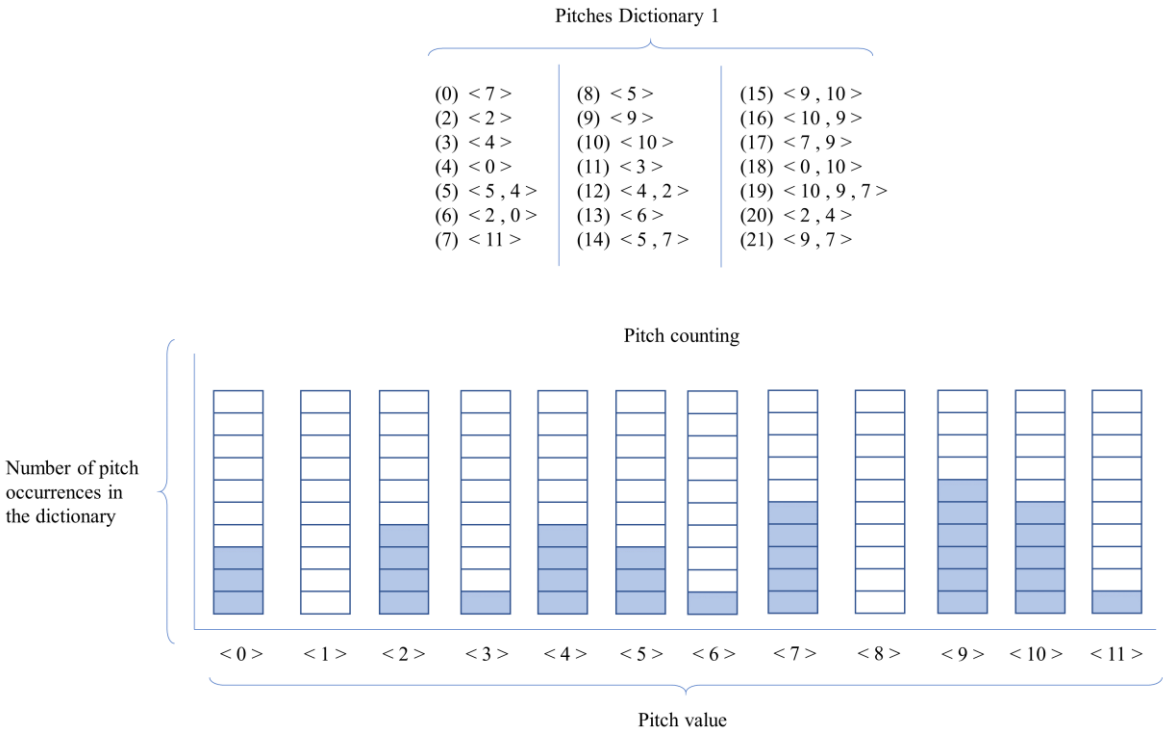


Figure 6-2: Result of counting the different pitches in *Dp1*. The *x-axis* corresponds to the pitch value, and the *y-axis* corresponds to the number of occurrences of each pitch value.

Then the vector is normalized by dividing each value in *H1* by *tp*. In this example *tp* = *33*. With Eq. (2). The vector is normalized.

$$\frac{H1}{tp} \tag{2}$$

The final result is $H1 = [.09, 0, .12, .03, .12, .09, .03, .15, 0, .18, .15, .03]$, which is the *global feature*. This vector has the values of the *global* feature of the dictionary *Dp1*. The next step is to analyze three histograms computed as the *local* features. The first step is to split the number of *words* in the dictionary into three. *Dp1* has a total amount of 21 *words* in it, so every division of the dictionary contains seven *words* in it to compute the histograms of each division *LH1*, *LH2*, and *LH3*, as depicted in Figure 6-3.

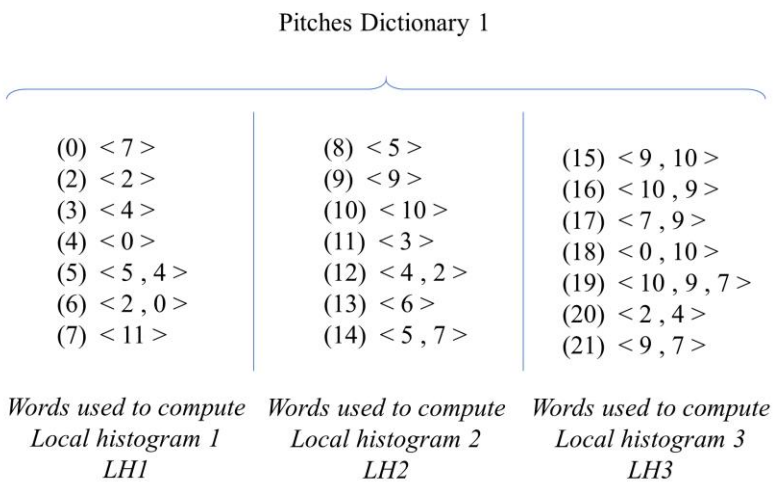


Figure 6-3: The three divisions of *DP1* and values in each of them to compute their histograms.

The following are the results of computing the local features of *DP1; tp(LH1)*=9, *tp*(*LH2)*=9, and *tp(LH3)*=15. *LH1=[2,0,2,0,2,1,0,1,0,0,0,1], LH2=[0,0,1,1,1,2,1,1,0,1,1,0], LH3=[1,0,1,0,1,0,0,3,0,5,4,0]* and the final normalized histograms are; *LH1=[.22,0,.22,0,.22,.11,0,.11,0,0,0,.11], LH2=[0,0,.11,.11,.11,.22,.11,.11,0,.11,.11,0], LH3=[.067, 0, .067, 0, .067, 0, 0, .2 ,0 , .33 , .26 , 0].*

By concatenating the *global* and *local* features vectors *H1, LH1, LH2, LH3*, a 48 values long vector is obtained. This vector is used in the fitness function to measure the fitness of each candidate generated using the original dictionary of pitches *Dp1* as the probability density function PDF.

Dp2 provides the needed constraints, *global* and *local,* to generate a structured dictionary to recombine with *C2* rhythmic dictionaries to generate the *C3*. The example of the structure is depicted in Figure 6-4.

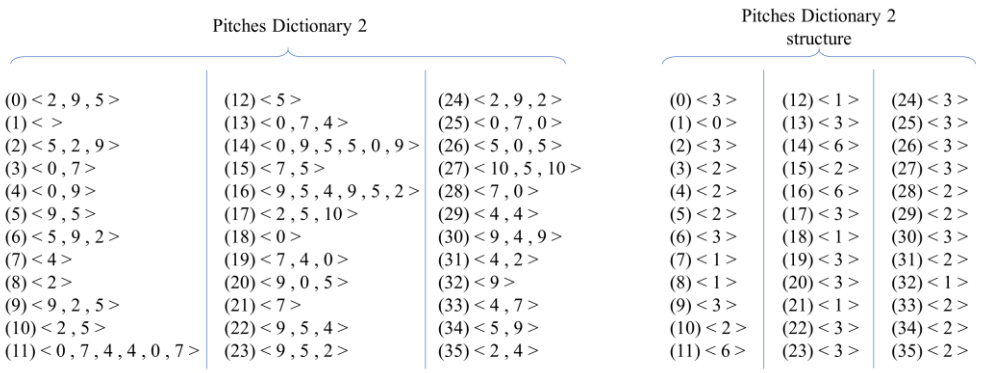


Figure 6-4: The original example of *Dp2* (left) and its structure, length of each *word* in it (right).

This dictionary structure provides us how many *words* we need to generate and the length of each of them. To start the generation, the original *DP1* is used as the probability density function, by picking values randomly from it. The following Figure 6-5 is a dictionary example generated from a population of 1000.

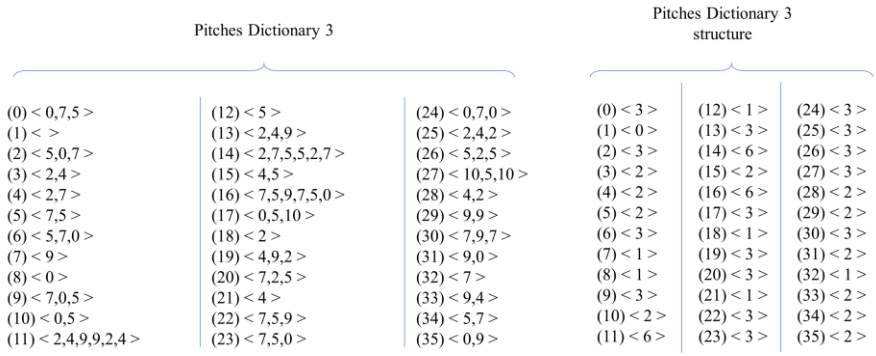


Figure 6-5: The generated dictionary *(left)* the structure of the dictionary *(right)*

The new dictionary with the correct structure has a *global H=[.14,0,.15,0,.10,.22,0,.2,0,.14,.03,0],* and the *local* histograms of *LH1=[.21,0,.14,0,.11,.21,0,.21,0,.11,0,0], LH2=[.09,0,.17,0,.11,.29,0,.2,0,.11,.03,0],* and *LH3=[.14,0,.14,0,.11,.14,0,.18,0,.21,.07,0].* The histograms generated in the *Dp3* generated a fitness value with the MSE of 0.005.

If a 1000 generations are done with PDF from the *seed* composition, the generated dictionaries are closer to the harmonic development; otherwise, they could be further away as depicted in Figure 6-6, and Figure 6-7.

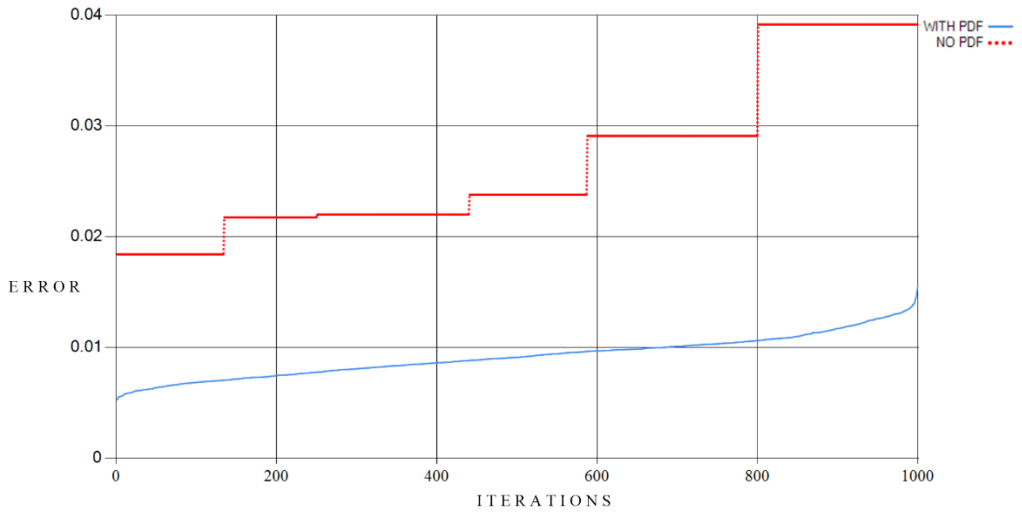


Figure 6-6: Original run of a 1000 sorted iterations, where the *x-axis* is the number of iterations and the *y-axis* the error measured by the MSE. In dotted-red the iterations with a random selection of values, in continuos-blue the generation using the *seed* dictionary.

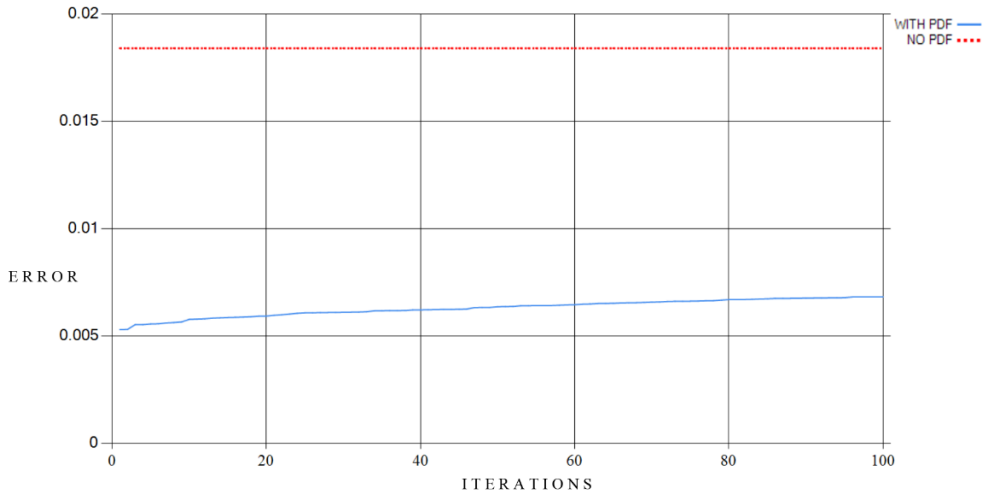


Figure 6-7: Original run of a 1000 sorted iterations with a *zoom* over the first 100 iterations, where the *x-axis* is the number of iterations and the *y-axis* the error measured by the MSE. In dotted-red the iterations with a random selection of values, in continuos-blue the generation using the *seed* dictionary.

In the following example, it was used Beethoven: Piano Sonata No. 14 Moonlight as the *rhythmic* section and Chopin: Trois Valses for the *harmonic* section to combine. The results show consistency in the rhythmic section without any change, and the consistency of the pitches, depicted in Figure 6-8, the first seven bars are shown as an excerpt from the result.

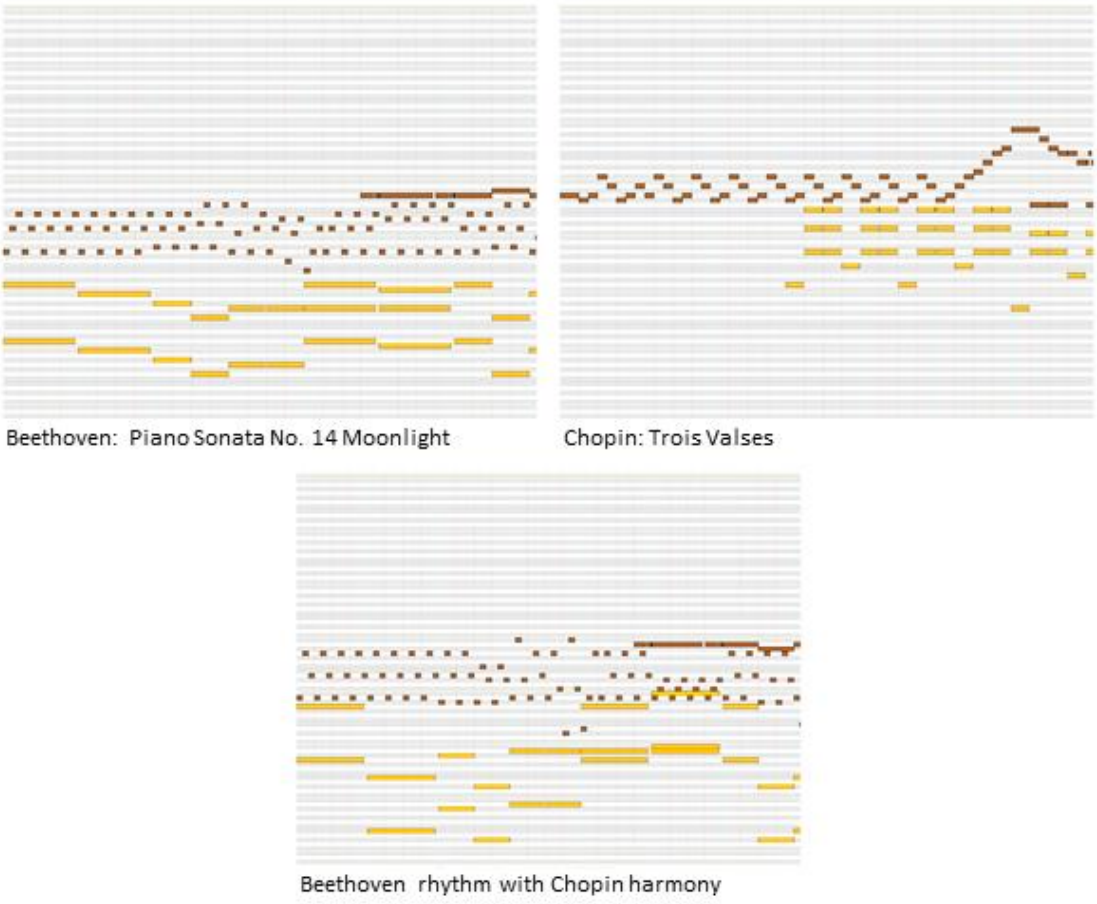


Figure 6-8: At the upper left side, we can observe the rhythmic patterns in Moonlight Sonata are kept on the result at the lower center side of the figure, it is combined with Trois Valses. Only the pitch dictionary was replaced, the octave dictionary was kept, we can observe how the notes are at the same place at the x-axis and changed on the pitch value on the y-axis.

The following example shows in Figure 6-9 how the dictionary replacement from the harmonic *seed* keeps the composition changing. Still, the rhythmic development maintains consistency in the onsets and the duration of the notes, holding the correctness of the difference between the rhythmic file and generated composition equal to zero.

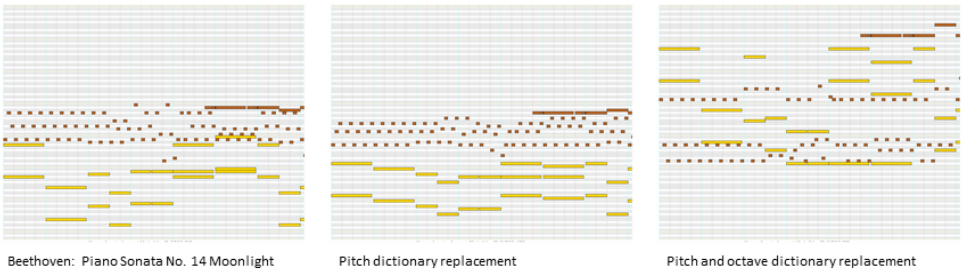


Figure 6-9: From left to right, the original Moonlight sonata MIDI file, at the center, pitch dictionary replacement and right, pitch and octave dictionary replacement result. The rhythmic development maintains consistency, and still, the composition keeps changing.

A sub excerpt from the previous example is depicted in Figure 6-10. Showing the first change of note value, then the change of that note, and at last how, when the original value in the original composition returns to the changed value.

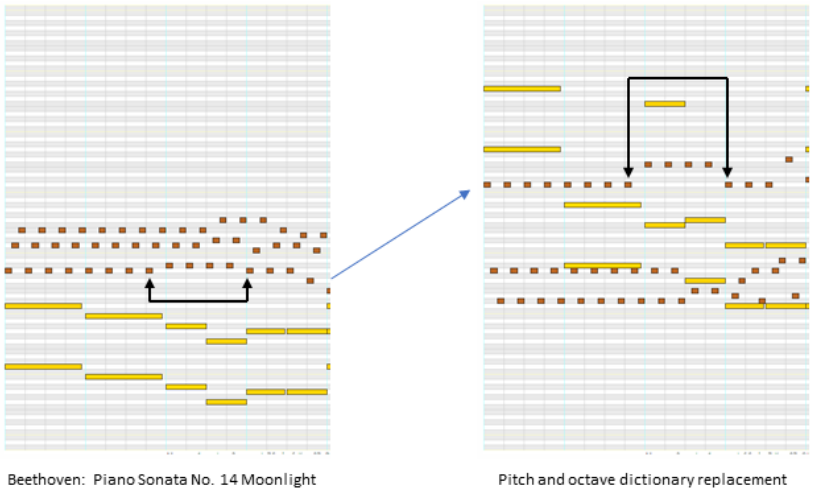


Figure 6-10: The original Moonlight sonata MIDI file(left), pointing to a pattern with the black arrows at the notes that change of value for a duration of four notes, and then they come back to the previous value. The transported values are shown after the dictionaries replacement of pitch and octave (right) shows the changing value and how it returns to the previously assigned value. This result is possible by encoding the dictionaries and ensures in case of motif that every time it is shown is going to be consistent with the changed value.

The generation of each dictionary has a constant value k since all of them have the same constraints, and the generation of the histograms also is a constant value h by counting each value in their dictionary. Then each vector of histograms is evaluated with a mean square

error (*MSE*), which is the squared sum of the difference of the vectors with a length equal to the size s of the histograms. Finally, to select the individual from the population, the list of the population is sorted by an incremental error from the desired histogram using quick sort algorithm which has a worst-case scenario of $\mathcal{O}(n^2)$ (Sedgewick, 1977). Therefore, the final complexity of the Dictionary generation and selection is equal to $\mathcal{O}(n^2 + (p * (k + h + s)))$

The obtained features with the proposed extraction and encoding were tested with Receiver Operating Characteristic (ROC) (Mandrekar, 2010). The implementation of the test was run on *scikit-learn*, which is a standard framework used for machine learning in *Python* (Pedregosa, 2011; Varoquaux, 2015). It was applied with ten-fold cross-validation on two sets of 15 elements each. Features from the elements were extracted with the proposed method from piano-sonatas of Beethoven and Bach fugues. The results of the classifiers are shown in Table 6-1, and the resulting confusion matrix of the best classifier is shown in Table 6-2, which validates the working designed method and resulting features from it.

Table 6-1. Results of Ten-fold cross-validation applied to different classifiers.

	Mean	**Variance**
GradientBoostingClassifier	0.7333	0.3266
RandomForestClassifier	0.8000	0.2211
LogisticRegression	0.8000	0.2211
PassiveAggressiveClassifier	**0.9000**	**0.1528**
SGDClassifier	0.8667	0.2211
SVC(linear)	0.8333	0.1667
RidgeClassifier	0.7333	0.2494
BaggingClassifier	0.8333	0.1667

Table 6-2. Confusion matrix from ten-fold cross-validation with Passive Aggressive classifier.

	Beethoven	**Bach**
Beethoven	14.0	1.0
Bach	2.0	13.0

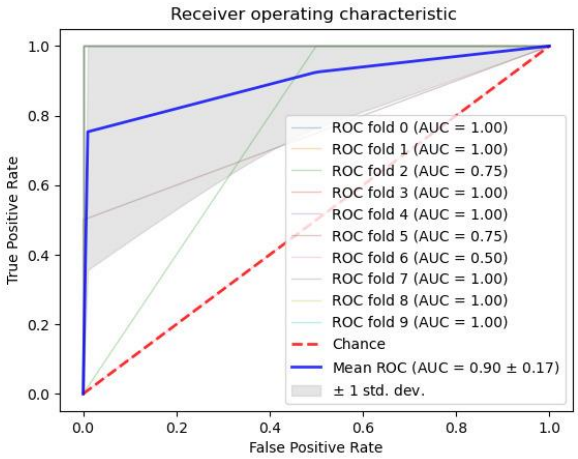


Figure 6-11: The Area Under the Curve (AUC) generated with the ten-fold classification of the Passive Aggressive classifier in the *scikit-learn* framework.

According to the tests and results of the Passive Aggressive classifier with total recognition of .9 and the Area Under the Curve performance test, the results are considered excellent (Mandrekar, 2010).

6.2 Discussion

A software was developed to read MIDI files, extract all the music bars according to the quarter note declared at the beginning of the files, to test each method. All the detailed processes of the methods were implemented to keep testing different approaches. The software and the results evolve during the exploration until the final model was completed.

At the new dictionary generation step, it is used a simplified version of a genetic algorithm reduced into only one population architecture. It generates mixtures from given examples like the nonlinear methods would combine like GANs and VAEs architectures but

without the need for the training from them and in a reduced number of examples (only two). The nonlinear methods with training make a higher complexity of implementation and runtime of these methods, which is reduced in our proposed method. That makes this model a better fit as a generative model when these are constrained.

The main difference in complexity with other methods is that the proposed one in this architecture does not need a large amount of data for abstraction and no training. Commonly the mentioned architectures have two-networks with more than two layers in each of the networks to cluster and abstract the information to start the generation of new information from the abstraction of the features in each dataset.

Even when the values from the generated population are low, this is because the histogram is normalized to one. That is why it is recommended to generate several individuals, and even the multiresolution of the histogram may vary. These can be adjusted according to what the user is trying to accomplish.

After analyzing the results obtained from the presented methods, some conclusions arise from the behavior of the newly generated data and the parameters of the recombination method. These outcomes should be taken into consideration when reproducing these methods, depending on each particular use from an experimental point of view or as a production and composition tool point of view.

The quality of the newly generated data depends on the quality of the original analyzed data for abstraction and recombination. Special events called *system events* in the MIDI file can interrupt or generate errors of continuity to recombine the files and generate new data, to avoid this error, a preprocessing step of cleaning data should be implemented to speed up the

results when experimenting. Since the MIDI file only supports 16 channels at a time. The method only works for a top number of 16 instruments. The method could have more than 16 channels, but there would have to be a number of virtual channels and split the MIDI file into those virtual channels.

Methods with neural networks need a limited size or duration for the generated file. Still, the encoding space to learn should be reduced, like the decoding of the octaves and pitches. These abstractions will help to generalize the learning space, and thus less training would be needed to be able to generate more general or different data. The duration of the selected files affects the resulting variations of the combinations by making parts harmonically repetitive when the harmonic MIDI is too short. There can be a measure to find how suitable are two files to be recombined or specify extensions for variation first of the shortest file.

When a file emulates several different instruments but all of them on one event channel, it is easier to control and maintain the complexity of the structures if it is generated a map for each instrument and share the same dictionaries. Either way, the dictionary will be the same used for all the instruments. It is as if two different music files share one set of dictionaries. This dictionary sharing reduces the indexes of structures and keeps working for the polyphony in both analysis and generation. The result is a composition music file that needs to be produced, which means that it must be transformed from the symbolic representation into the signal representation. It could be recorded with players or using virtual instruments. The results can work as both templates of complete music compositions and generate ideas of variations from music files to use them in pieces or as a whole.

In appendix C, further examples of rhythmic patterns with different harmonic features are shown.

Chapter 7 Contributions, Conclusions, and future work

This chapter summarizes the essential ideas of this research and tries to mention the most critical aspects of global conclusions. These details aim to find a bigger goal to increment different possible approaches for further research presenting new objectives and possible solutions for such explorations.

7.1 Contributions

The following papers, chapters, and workshops were published as a result of the research done in this thesis. They are directly related to the presented solution found by the experiments and implementations.

***Conferences*:**

The following paper, “Algorithmic music composition based on artificial intelligence: A survey,” was written to be presented at the CONIELECOMP conference. It presents related work found by making a review of the state of the art of the problem, and they were classified according to the area and subarea of each used methodology. (Lopez-Rincon et al., 2018b)

Book Chapter:

The following chapter was written as a summary of outreach. The importance of this chapter is that it reflects the ability to explain the ideas outside the academic area. It is a resume of the article “Music

Visualization Based on Spherical Projection with Adjustable Metrics" but with a common language without technical terms to be read by the community outside the academia. (Lopez-Rincon & Starostenko, 2019)

Workshops:

These papers were presented at the workshop of Technology, Science and Culture, A Global Vision.

In "A 3D Spatial Visualization of Measures in Music Compositions", presented a method for dimensionality reduction of MIDI file by extracting the events found in the file structure and subdivide them as a signal but proposing a windowing technique found in the file as music theory. This idea evolved to produce the paper "Music Visualization Based on Spherical Projection with Adjustable Metrics," which found a self-contained descriptor in each of the MIDI files. (Lopez-Rincon & Starostenko, 2018)

It was designed as a one network generative model called "A Simplified Generative Model Based on Gradient descent and Mean Square Error." It reduces the complexity of training and implementation in half, and it can be used in small generations. For a more complex generation, a change in the error metric should be designed to be able to refine peaks of changes and not only average the result of them.(Lopez-Rincon & Starostenko, 2019)

Journals

The methodology described on how to analyze MIDI files with self-reference for auto adjustment and customizable metrics for visualization analysis was presented in (Lopez-Rincon & Starostenko, 2019a)**.** This paper is the first outcome of the research on how to analyze the music composition file by normalizing its data. This analysis is achieved by finding sub-features and normalize them for later use.

The idea of the sub-features analysis, and the use of them to recognize a context was proven in (Rosas-Romero et al., 2019), were a classifier pixel by pixel was developed. The same idea of the sub-features to classify a value by its context applied in classification in computer vision.

As the hypothesis stated, it was proven that it is possible to extract features from already existing compositions and use them to create new compositions that also share such features. It was designed a methodology to analyze music data with no labels and extract from it the main characteristics of a composition. Using these features for a rapid prototype of compositions is finished as fast as even for interaction time. This methodology uses a symbolic representation of the music elements in MIDI files.

The quarter note descriptor in the MIDI file allows the architecture to quantize the music events in the data and group them into the abstraction of *words* found in every music file to generate a dictionary because all the events are related to this unique descriptor. It was possible to group the musical elements as harmonic and rhythmic and then subdivide them again into *pitch, octave, onset,* and *duration.* Using this encoding of the elements allowed to recombine them from different compositions to create new music declared in a MIDI file structure of events. This encoding was achieved, given a visualization tool developed during the exploration phase of the thesis.

7.2 Conclusions

This research focused on experiments based on MIDI files, which have a structure to represent in discrete values the symbolic representation of sounds developing in time. These files were obtained from paid databases and free files from different websites. Regarding the available files, there are plenty of repositories and databases, but there are no standard definitions other than the file structure. Most electronic instruments and tools like digital audio workstations (DAW) have support for MIDI protocol and the MIDI file structure. This broad support is part of its popularity and availability but, at the same time, presents a challenge to find common values inside the symbolic descriptions to normalize them and thus be able to process them. Besides the challenge of finding a structure to normalize the representation of the files regardless of their different sizes and different resolutions, there were other objectives of the research as it was designing methods for solving specific tasks in the analysis.

The proposed method determines different kinds of features used for visualizations to choose one that better fit to explore a specific problem. Mainly, it has been exploited for the tonal representation of music compositions in 3D plots. A self-similarity matrix SSM based on the Euclidian and Hamming distances between vectors in 12-dimensional space has been proposed to use. Smooth and decreasing-only behavior of adjustment average error AAE, which relates dimensionality reduction errors with iterations during setting vectors in 3D space, has been adopted as a stop criterion of the algorithm for adjustment of vector positions. It facilitates finding a threshold for the projection of reduced spatial dimensionality with an error of less than 10% after only 20 iterating cycles.

The Hamming distance is quite simple to use for practical implementation due to its binary representation. In contrast, for visualization, the Euclidean distance provides a scaled result

proportional to the square root. Both provide a similar spatial but scaled vector magnitude projections. The main difference is the more straightforward implementation of the Hamming distance and, as a consequence, the higher speed of dimensionality reduction.

It was discovered that the autoencoder is faster because it needs fewer epochs to converge. Still, it reduces dimensionality redundancy using entirely numerical analysis based on nonlinear relation inside the neural network. It also has a stop criterion, but the reduction is not based on specific metrics designed for data analysis or for interpreting the music data. The proposed method can make low dimensional projection and expose visual patterns applying the same process for operating with different parameters used for performance analysis. Additionally, it considers the silence as a vector, and all the relations of the vectors are set around it. No other methods have been found that consider silence; however, it is an essential part of the music composition process in general. Mainly, silence is considered as the current starting point for a music piece and must be counted before sound begins.

The theoretical contributions consist of a flexible methodology tool to analyze MIDI files, which represent music as a list of events. This analysis is done by using the crotchet declared in the MIDI file as a self-reference to set a size of a window time to split and analyze music compositions. Then the reduction of all possible combinations of harmonies in 12D space is applied to convert them into a finite set of 3D vectors, which can be used for analysis and visualization of harmonic features in a file. The practical contributions include an original technique developed for the projection of music composition features in 3-dimensional space based on a feature computed as a self-similarity matrix from higher dimensional vectors.

The second theoretical contribution was a complete method for analysis and recombination of features in MIDI files entirely autonomous with no human interaction during the

compositional process or at the evaluation steps of them. It consists of a complete process of new music data generation based on the recombination of harmonic development of a composition adapted to the rhythmic development of a second composition. This recombination is achieved by a designed abstraction of sub-features to generate dictionaries containing different combinations of values found in excerpts of music compositions, assigning an indexed value to these combinations. Then a genetic algorithm was adapted to be used to abstract the sub-features of one composition and adjust them to work with the features of another composition. By recombining the adjusted features, a complete polyphonic and poly instrumental composition is generated.

7.3 Future Work

Different approaches of composition music generation were analyzed, and two methods were proposed, these models of analysis and music recombination are the first results of the process of research and experiments with different tools to find these fundamental methodologies. Automatic data generation and artificial intelligence, in general, is a growing area with new results every month. These approaches were able to tackle the characteristics of data analysis with no labeled information to learn the main features that describe them and to propose manners to analyze them automatically and generate new data with such features.

As future work, the development of the generative model could benefit from the creation of the profiles of songs based on the proposed method for extraction and projection of data with a reduced computational cost. A weighted version of the time windows at each moment could be developed, taking into consideration the complete sequence of the tones using as

metric the cosine distance to generate the SSM. This abstraction can help to precise better the composition profile. The profile could be used to design an objective function by finding the global optima exploiting meta-heuristics (Liu, Y., Pan, J. & Su, Z. 2017) like genetic algorithm or simulated annealing. Both the profile and meta-heuristics can serve to implement a more efficient generative model. Also, with this method, it is possible to create a composer recognition approach based on dimensionality reduction of harmonics. We already have some promising results with this new approach.

The normalized values in the MIDI events can be used to represent complete composition as a metric of the harmonic content of a song. Using them as a histogram of the whole and reducing it into a 12-dimensional array of continuous values, they can be the representative vectors of each song to classify classical composers. There is even the possibility to use different subdivisions of the composition to compare among other composers. These subdivisions can be used in a fitness function as a metric after training, to evaluate the harmonic content of a composition: human, computer, or both.

The subdivision of the symbolic composition could be translated the same way into the signal side of the music, by using tools like wavelets or window short time Fourier, an analysis of the self-declared beat in the composition and change from time domain the signal to find thresholds of differences from excerpts of audio.

Another possible continuation of these methods is the possibility of the abstraction of the rhythmic statistical behavior. The attempt should again try to obtain results at different dictionary resolutions to obtain a rhythmic development, preserving polyphony and not constrained to composition duration other than the imposed by the user.

Finally, possible exciting research could be the possibility of retargeting the main ideas from these methods to reuse them in other areas or with different file structures also to extract the elements that form those files and generate and recombine their features.

References

Ackerman, M., & Loker, D. (2016). Algorithmic Songwriting with ALYSIA. *CoRR, abs/1612.01058*. http://arxiv.org/abs/1612.01058

AIVA. (2020). *AIVA - Artificial Intelligence Virtual Artist, The Artificial Intelligence composing emotional soundtrack music*. https://www.aiva.ai/

Aloupis, G., Fevens, T., Langerman, S., Matsui, T., Mesa, A., Nuñez, Y., Rappaport, D., & Toussaint, G. (2006). Algorithms for Computing Geometric Measures of Melodic Similarity. *Computer Music Journal*, *30*(3), 67–76. https://doi.org/10.1162/comj.2006.30.3.67

Back, D. (2020, February). Standard MIDI-File Format Spec. 1.1, updated. *Standard MIDI-File Format Spec. 1.1*. http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfile format.html

Bergstrom, T., Karahalios, K., & Hart, J. C. (2007). Isochords: Visualizing Structure in Music. *Proceedings of Graphics Interface 2007 on - GI '07*, 297.

Bickerman, G., Bosley, S., Swire, P., & Keller, R. M. (2010). Learning to Create Jazz Melodies Using Deep Belief Nets. *First International Conference on Computational Creativity*, 14.

Bigo, L., & Conklin, D. (2016). A Viewpoint Approach to Symbolic Music Transformation. In R. Kronland-Martinet, M. Aramaki, & S. Ystad (Eds.), *Music, Mind, and*

Embodiment (Vol. 9617, pp. 213–227). Springer International Publishing. https://doi.org/10.1007/978-3-319-46282-0_13

Blier, L., Wolinski, P., & Ollivier, Y. (2018). Learning with Random Learning Rates. *ArXiv:1810.01322 [Cs, Stat]*. http://arxiv.org/abs/1810.01322

Boden, M. A., & Edmonds, E. A. (2009). What is generative art? *Digital Creativity*, *20*(1–2), 21–46.

Bretan, M., Weinberg, G., & Heck, L. (2016). *A Unit Selection Methodology for Music Generation Using Deep Neural Networks*. arXiv preprint arXiv:1612.03789.

Brunner, G., Konrad, A., Wang, Y., & Wattenhofer, R. (2018). MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer. *Proc. ISMIR Int. Soc. Music Infor. Retr. Conf.*, 1–8. http://arxiv.org/abs/1809.07600

Cambouropoulos, E., Kaliakatsos-Papakostas, M. A., & Tsougras, C. (2014). An idiom-independent representation of chords for computational music analysis and generation. *ICMC*. https://www.researchgate.net/profile/Emilios_Cambouropoulos/publication/266614715_An_Idiom-independent_Representation_of_Chords_for_Computational_Music_Analysis_and_Generation/links/54354b240cf2bf1f1f286e3e.pdf

Chen, C.-H., Weng, M.-F., Jeng, S.-K., & Chuang, Y.-Y. (2008). Emotion-Based Music Visualization Using Photos. In S. Satoh, F. Nack, & M. Etoh (Eds.), *Advances in Multimedia Modeling* (pp. 358–368). Springer Berlin Heidelberg.

Chew, E., & Chen, Y.-C. (2005). Real-Time Pitch Spelling Using the Spiral Array. *Computer Music Journal*, *29*(2), 61–76. https://doi.org/10.1162/0148926054094378

Chu, H., Urtasun, R., & Fidler, S. (2016). *SONG FROM PI: A MUSICALLY PLAUSIBLE NETWORK FOR POP MUSIC GENERATION*. ICLR.

Ciuha, P., Klemenc, B., & Solina, F. (2010). Visualization of concurrent tones in music with colours. *Proceedings of the International Conference on Multimedia - MM '10*, 1677. https://doi.org/10.1145/1873951.1874320

Cohn, R. (1998). *Introduction to Neo-Riemannian Theory: A Survey and a Historical Perspective* (http://www.jstor.org/stable/843871). *42*(2), 167–180. https://doi.org/10.2307/843871

Cox, G. (2010). On the Relationship Between Entropy and Meaning in Music: An Exploration with Recurrent Neural Networks. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *32*, 6.

Dreier, J. (2015). *Algorithmic Music Composition*. 6.

Eigenfeldt, A., Bown, O., Brown, A. R., & Gifford, T. (2016). Flexible Generation of Musical Form: Beyond Mere Generation. *Seventh International Conference on Computational Creativity*, 8.

Eigenfeldt, A., Bown, O., R. Brown, A., & Gifford, T. (2016). *Flexible Generation of Musical Form: Beyond Mere Generation*. Proceedings of the Seventh International Conference on Computational Creativity.

Eigenfeldt, A., & Pasquier, P. (2012). Creative Agents, Curatorial Agents, And Human-Agent Interaction In Coming Together. *9th Sound and Music Computing Conference*, 181–186. https://doi.org/10.5281/ZENODO.850074

Eppe, M., Confalonieri, R., Maclean, E., Kaliakatsos, M., Cambouropoulo, E., Schorlemmer, M., Codescu, M., & Kühnberger, K.-U. (2015). *Computational invention of cadences*

and chord progressions by conceptual chord-blending. International Joint Conferences on Artificial Intelligence.

Febres, G., & Jaffe, K. (2017). Music viewed by its entropy content: A novel window for comparative analysis. *PLOS ONE*, *12*(10), e0185757. https://doi.org/10.1371/journal.pone.0185757

Fonteles, J. H., Rodrigues, M. A. F., & Basso, V. E. D. (2013). Creating and evaluating a particle system for music visualization. *Journal of Visual Languages & Computing*, *24*(6), 472–482. https://doi.org/10.1016/j.jvlc.2013.10.002

Foote, J., & Cooper, M. (2001). *Visualizing Musical Structure and Rhythm via Self-Similarity*. 4.

Fukayama, S., Nakatsuma, K., Sako, S., Nishimoto, T., & Sagayama, S. (2010). Automatic song composition from the lyrics exploiting prosody of Japanese language. *7th Sound and Music Computing Conference*, 4. https://zenodo.org/record/849727#.XlVXQ6hKhPY

Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumeé III, H., & Crawford, K. (2018). Datasheets for Datasets. *ArXiv:1803.09010 [Cs]*. http://arxiv.org/abs/1803.09010

Ghisi, D. (2017). *Music across music: Towards a corpus-based, interactive computer-aided composition*. Université Pierre et Marie Curie.

Google, A. (2019, February). *Magenta – open-source research project*. https://ai.google/research/teams/brain/magenta/

Grachten, M., & Chacón, C. E. C. (2017). Strategies for Conceptual Change in Convolutional Neural Networks. *CoRR, abs/1711.01634*. http://arxiv.org/abs/1711.01634

Hadjeres, G., & Pachet, F. (2016). *DeepBach: A Steerable Model for Bach chorales generation*. arXiv preprint arXiv:1612.01010.

Hailesilassie, T. (2016). Rule Extraction Algorithm for Deep Neural Networks: A Review. *International Journal of Computer Science and Information Security*, *14*(7), 6.

He, Y.-H. (2017). A Visualization of the Classical Musical Tradition. *ArXiv:1709.04038 [Stat]*, 1–9.

Helber, P., Bischke, B., Dengel, A., & Borth, D. (2017). EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification. *ArXiv:1709.00029 [Cs]*. http://arxiv.org/abs/1709.00029

Herremans, D., & Chew, E. (2019). MorpheuS: Generating Structured Music with Constrained Patterns and Tension. *IEEE Transactions on Affective Computing*, *10*(4), 510–523. https://doi.org/10.1109/TAFFC.2017.2737984

Herremans, D., & Chew, E. (2016). Tension ribbons: Quantifying and visualizing tonal tension. *Music Notation and Representation (TENOR)*, 10.

Hinton, G. E. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hossain, MD. Z., Sohel, F., Shiratuddin, M. F., & Laga, H. (2019). A Comprehensive Survey of Deep Learning for Image Captioning. *ACM Computing Surveys*, *51*(6), 1–36. https://doi.org/10.1145/3295748

Huang, A., & Wu, R. (2016). Deep Learning for Music. *ArXiv Preprint ArXiv:1606.04930*. http://arxiv.org/abs/1606.04930

Johnson, D. D. (2017). Generating Polyphonic Music Using Tied Parallel Networks. In J. Correia, V. Ciesielski, & A. Liapis (Eds.), *Computational Intelligence in Music*,

Sound, Art and Design (Vol. 10198, pp. 128–143). Springer International Publishing. https://doi.org/10.1007/978-3-319-55750-2_9

Kaliakatsos–Papakostas, M. A., Epitropakis, M. G., Floros, A., & Vrahatis, M. N. (2012). *Interactive evolution of 8-bit melodies*. International Conference on Evolutionary and Biologically Inspired Music and Art, Berlin Heidelberg.

Kaliakatsos-Papakostas, M. A., Epitropakis, M. G., & Vrahatis, M. N. (2011). Weighted Markov Chain Model for Musical Composer Identification. In C. Di Chio, A. Brabazon, G. A. Di Caro, R. Drechsler, M. Farooq, J. Grahl, G. Greenfield, C. Prins, J. Romero, G. Squillero, E. Tarantino, A. G. B. Tettamanzi, N. Urquhart, & A. Ş. Uyar (Eds.), *Applications of Evolutionary Computation* (Vol. 6625, pp. 334–343). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-20520-0_34

Kaski, S., & Peltonen, J. (2011). Dimensionality Reduction for Data Visualization [Applications Corner]. *IEEE Signal Processing Magazine*, *28*(2), 100–104. https://doi.org/10.1109/MSP.2010.940003

Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. *Proc. Int. Conf. Learn. Repres.*, 1–14. http://arxiv.org/abs/1312.6114

Korzeniowski, F., & Widmer, G. (2018, August 15). Genre-Agnostic Key Classification with Convolutional Neural Networks. *ArXiv:1808.05340 [Cs, Eess]*. 19th Int. Soc. for Music Inf. Retrieval Conference, Paris, France. http://arxiv.org/abs/1808.05340

KS, K., & Sangeetha, S. (2019). SECNLP: A Survey of Embeddings in Clinical Natural Language Processing. *ArXiv:1903.01039 [Cs]*. http://arxiv.org/abs/1903.01039

Kunimatsu, K., Ishikawa, Y., Takata, M., & Joe, K. (2015). *A Music Composition Model with Genetic Programming-A Case Study of Chord Progression and Bassline*. The

Steering Committee of The World Congress in Computer Science, Computer Engineering, and Applied Computing (WorldComp).

Kunstderfuge. (2019). *Kunstderfuge, Classical Music*.

Li, X. R., & Zhao, Z. (2001a). *Measures of performance for evaluation of estimators and filters* (O. E. Drummond, Ed.; p. 530). https://doi.org/10.1117/12.492751

Li, X. R., & Zhao, Z. (2001b). *Measures of performance for evaluation of estimators and filters* (O. E. Drummond, Ed.; p. 530). https://doi.org/10.1117/12.492751

Liu, Y., Pan, J., & Su, Z. (2017). Deep Blind Image Inpainting. *International Conference on Intelligent Science and Big Data Engineering*, 128–141. http://arxiv.org/abs/1712.09078

Livingstone, S. R., Muhlberger, R., Brown, A. R., & Thompson, W. F. (2010). Changing Musical Emotion: A Computational Rule System for Modifying Score and Performance. *Computer Music Journal*, *34*(1), 41–64. https://doi.org/10.1162/comj.2010.34.1.41

Logan, M. (2020, May 4). *Recent Developments with the MIDI Standard- Finale*. https://www.finalemusic.com/blog/recent-developments-midi-standard/

Lopez, R., Regier, J., Jordan, M. I., & Yosef, N. (2018). Information Constraints on Auto-Encoding Variational Bayes. *ArXiv:1805.08672 [Cs, q-Bio, Stat]*. http://arxiv.org/abs/1805.08672

Lopez-Rincon, O., & Starostenko, O. (2019a). Music Visualization Based on Spherical Projection With Adjustable Metrics. *IEEE Access*, *7*, 140344–140354. https://doi.org/10.1109/ACCESS.2019.2944083

Lopez-Rincon, O., & Starostenko, O. (2019b, October 11). *A Simplified Generative Model Based on Gradient Descent and Mean Square Error*. https://doi.org/10.5772/intechopen.90099

Lopez-Rincon, O., & Starostenko, O. (2019c). Creando un creador: Visualización de armonías en composiciones musicales. In *La velocidad de la pausa*. UDLAP.

Lopez-Rincon, O., & Starostenko, O. (2018). A 3D Spatial Visualization of Measures in Music Compositions. *Technology, Science, and Culture - A Global Vision*, *1*, 119. https://doi.org/10.5772/intechopen.83691

Lopez-Rincon, O., Starostenko, O., & Martin, G. A.-S. (2018a). Algorithmic music composition based on artificial intelligence: A survey. *2018 International Conference on Electronics, Communications, and Computers (CONIELECOMP)*, 187–193. https://doi.org/10.1109/CONIELECOMP.2018.8327197

Lopez-Rincon, O., Starostenko, O., & Martin, G. A.-S. (2018b). Algorithmic music composition based on artificial intelligence: A survey. *2018 International Conference on Electronics, Communications, and Computers (CONIELECOMP)*, 187–193. https://doi.org/10.1109/CONIELECOMP.2018.8327197

Lyu, Q., Wu, Z., Zhu, J., & Meng, H. (2015). *Modeling High-Dimensional Sequences with LSTM-RTRBM: Application to Polyphonic Music Generation*. Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence.

Maaten, L. van der. (2014). Accelerating t-SNE using Tree-Based Algorithms. *Journal of Machine Learning Research*, *15*, 3221–3245.

Makris, D., Kaliakatsos-Papakostas, M. A., & Cambouropoulos, E. (2015). *PROBABILISTIC MODULAR BASS VOICE LEADING IN MELODIC HARMONISATION*. ISMIR.

Makris, D., Maximos A. Kaliakatsos-Papakostas, & Emilios Cambouropoulos. (2015). Probabilistic Modular Bass Voice Leading In Melodic Harmonisation. *Proceedings of the 16th International Society for Music Information Retrieval Conference,* 323–329. https://doi.org/10.5281/ZENODO.1416374

Mandrekar, J. N. (2010). Receiver Operating Characteristic Curve in Diagnostic Test Assessment. *Journal of Thoracic Oncology*, *5*(9), 1315–1316. https://doi.org/10.1097/JTO.0b013e3181ec173d

Maximos Kaliakatsos, – Papakostas, & Emilios Cambouropoulos. (2014, September 14). Probabilistic harmonization with fixed intermediate chord constraints. *Joint 40th International Computer Music Conference (ICMC) and 11th Sound and Music Computing (SMC) Conference (ICMC-SMC2014)*. https://doi.org/10.13140/2.1.3079.5526

McCaig, G., DiPaola, S., & Gabora, L. (2016). Deep convolutional networks as models of generalization and blending within visual creativity. *ArXiv Preprint ArXiv:1610.02478*. https://arxiv.org/abs/1610.02478

McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and Music Signal Analysis in Python. *Proceedings of the 14th Python in Science Conference*, *8*, 18–24. https://doi.org/10.25080/Majora-7b98e3ed-003

Meredith, D. (2014). Compression-based geometric pattern discovery in music. *2014 4th International Workshop on Cognitive Information Processing (CIP)*, 1–6. https://doi.org/10.1109/CIP.2014.6844503

Navarro, M., Corchado, J., & Demazeau, Y. (2014). A Musical Composition Application Based on a Multiagent System to Assist Novel Composers. *Fifth International Conference on Computational Creativity*, 5.

Nierhaus, G. (2009). *Algorithmic Composition Paradigms of Automated Music Generation*. SpringerWien New York.

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., & Walsh, J. (2020). Deep Learning vs. Traditional Computer Vision. In K. Arai & S. Kapoor (Eds.), *Advances in Computer Vision* (Vol. 943, pp. 128–144). Springer International Publishing. https://doi.org/10.1007/978-3-030-17795-9_10

Pachet, F., Roy, P., & Barbieri, G. (2011). Finite-Length Markov Processes with Constraints. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 8.

Papadopoulos, A., Roy, P., & Pachet, F. (2016). *Assisted Lead Sheet Composition using FlowComposer*. International Conference on Principles and Practice of Constraint Programming.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.*, *12*(null), 2825–2830.

Prince, J. B. (2014). Contributions of pitch contour, tonality, rhythm, and meter to melodic similarity. *Journal of Experimental Psychology: Human Perception and Performance*, *40*(6), 2319–2337. https://doi.org/10.1037/a0038010

Quick, D. (2016). Learning production probabilities for musical grammar. *Journal of New Music Research*, *45*(4), 295–313. https://doi.org/10.1080/09298215.2016.1228680

Quick, D., & Hudak, P. (2013). A temporal generative graph grammar for harmonic and metrical structure. *ICMC*. https://www.researchgate.net/profile/Donya_Quick/publication/281557620_A_Tem

poral_Generative_Graph_Grammar_for_Harmonic_and_Metrical_Structure/links/55edc1fa08ae0af8ee1935d8.pdf

Rosas-Romero, R., Lopez-Rincon, O., & Starostenko, O. (2019). Fully automatic alpha matte extraction using artificial neural networks. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-019-04154-4

Sapp, C. S. (2001). Harmonic Visualizations of Tonal Music. *Proc. Int. Comp. Music Conf*, *2001*, 8–18.

Scirea, M., Barros, G. A., Shaker, N., & Togelius, J. (2015). SMUG: Scientific music generator. *Proceedings of the Sixth International Conference on Computational Creativity June*, 204. http://axon.cs.byu.edu/ICCC2015proceedings/9.2Scirea.pdf

Scirea, M., Togelius, J., Eklund, P., & Risi, S. (2016). *MetaCompose: A Compositional Evolutionary Music Composer*. International Conference on Evolutionary and Biologically Inspired Music and Art.

Sedgewick, R. (1977). The analysis of Quicksort programs. *Acta Informatica*, *7*(4), 327–355. https://doi.org/10.1007/BF00289467

Snydal, J., & Hearst, M. (2005). ImproViz: Visual Explorations of Jazz Improvisations. *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, 1805.

Stolzenburg, F. (2015). Harmony perception by periodicity detection. *Journal of Mathematics and Music*, *9*(3), 215–238. https://doi.org/10.1080/17459737.2015.1033024

Suits, B. (2020, February). Frequencies for equal-tempered scale, A4 = 440 Hz. *Frequencies for Equal-Tempered Scale, A4 = 440 Hz*. https://pages.mtu.edu/~suits/notefreqs.html

Theis, L., Shi, W., Cunningham, A., & Huszár, F. (2017). Lossy Image Compression with Compressive Autoencoders. *ArXiv:1703.00395 [Cs, Stat]*. http://arxiv.org/abs/1703.00395

ThinkSync Music. (2020, May 1). *Budgeting Consideration When Using Music in Films—ThinkSync Music*. http://thinksyncmusic.com/guides/music-in-films-budgeting-continued/

Toussaint, G. (2010). Computational geometric aspects of rhythm, melody, and voice-leading. *Computational Geometry*, *43*(1), 2–22. https://doi.org/10.1016/j.comgeo.2007.01.003

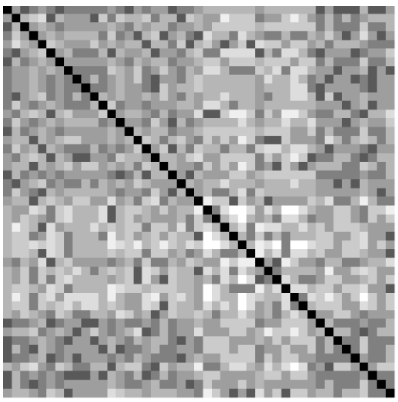
Varoquaux, G., Buitinck, L., Louppe, G., Grisel, O., Pedregosa, F., & Mueller, A. (2015). Scikit-learn: Machine Learning Without Learning the Machinery. *GetMobile: Mobile Computing and Communications*, *19*(1), 29–33. https://doi.org/10.1145/2786984.2786995

Warren, H. S. (2013). *Hacker's delight* (2nd ed). Addison-Wesley.

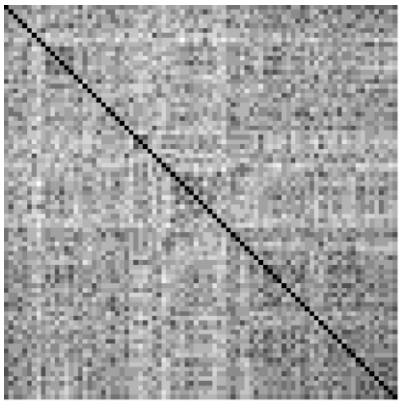
Wolkowicz, J., Brooks, S., & Keselj, V. (2009). MIDIVIS: Visualizing Music Pieces Structure via Similarity Matrices. *ICMC*, 53-56. https://pdfs.semanticscholar.org/ceda/27fd01f945970bbe6ebacf98ce7a05ca8120.pdf?_ga=2.147966260.280370585.1564720651-1619311515.1527244310

Yu, L., Zhang, W., Wang, J., & Yu, Y. (2017). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *ArXiv:1609.05473 [Cs]*, *4*. http://arxiv.org/abs/1609.05473

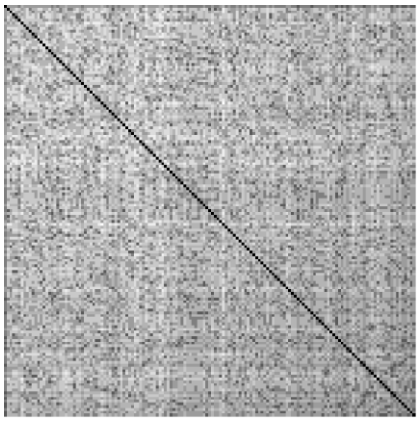
Appendix A Examples of different SSM feature extractions



Ave Maria by Bach

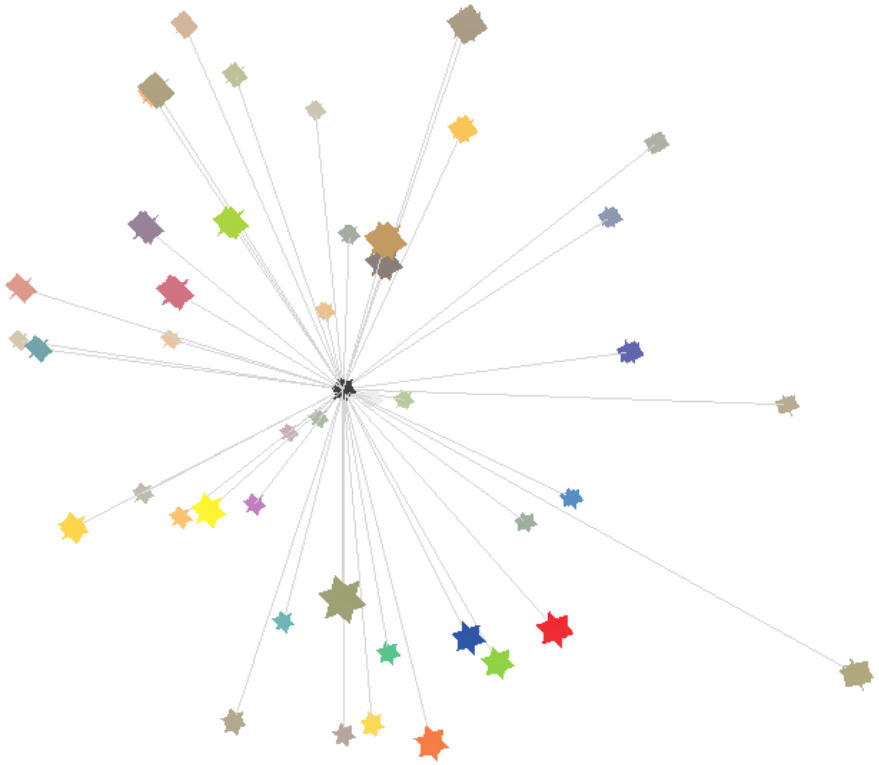


Fur Elise by Beethoven

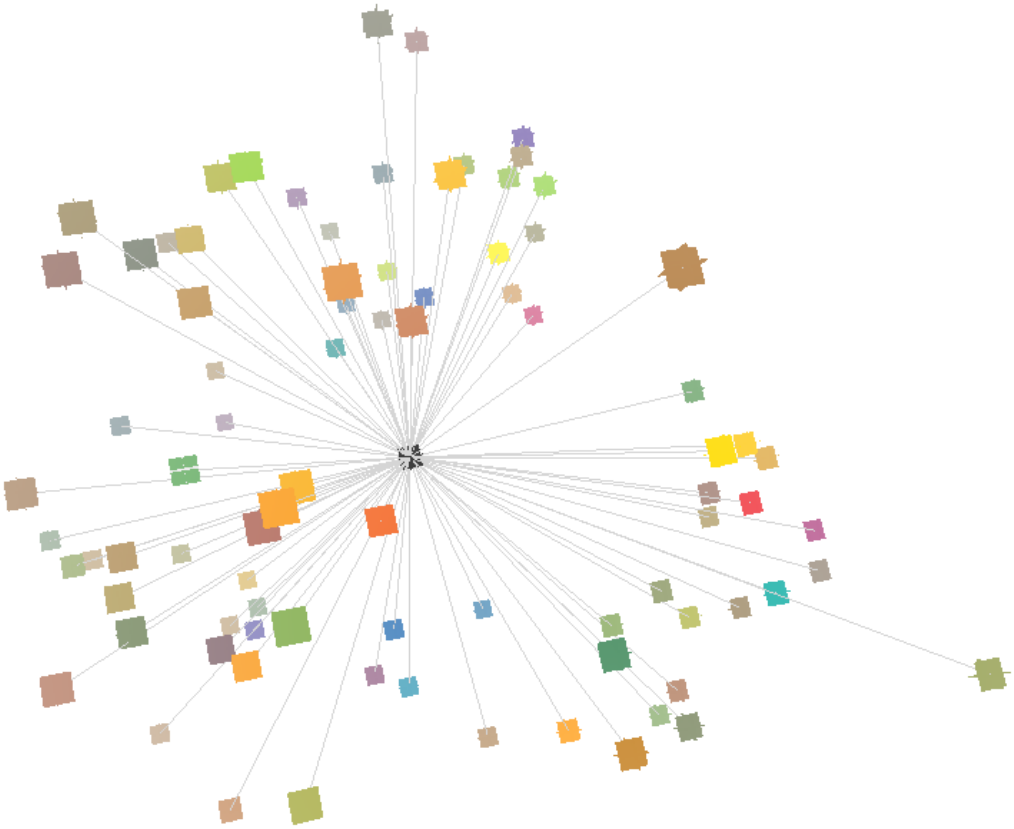


Trois Valses by Chopin

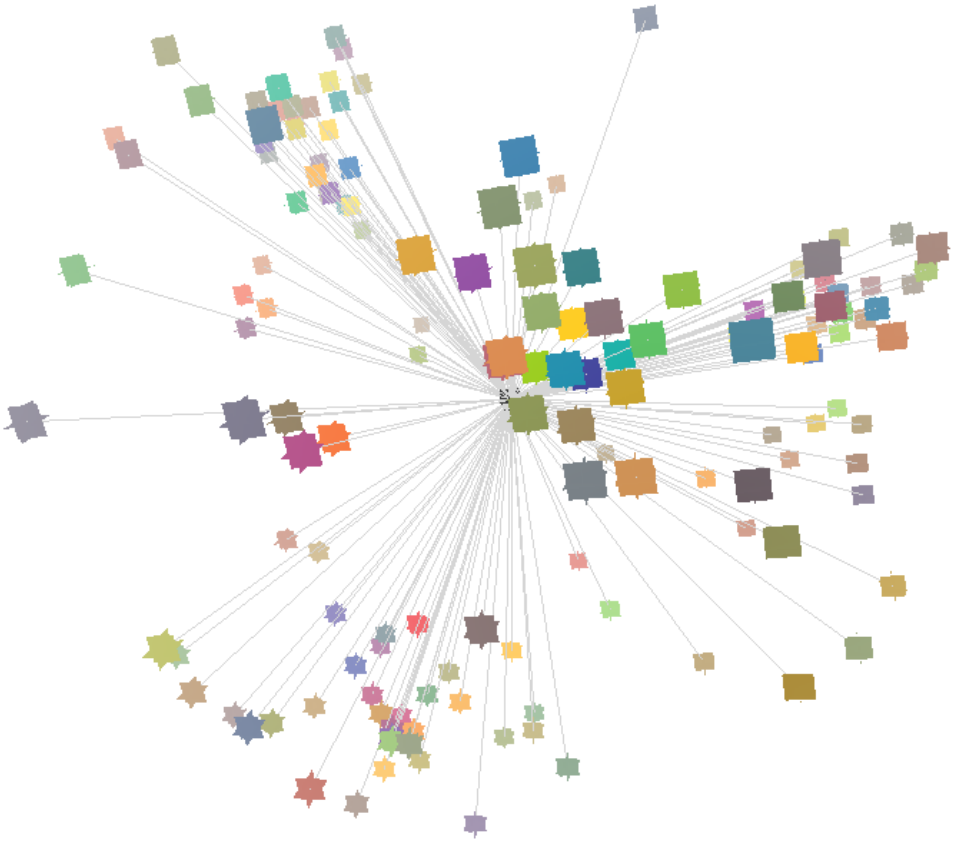
Appendix B Examples of different spherical projections



Bach Ave Maria



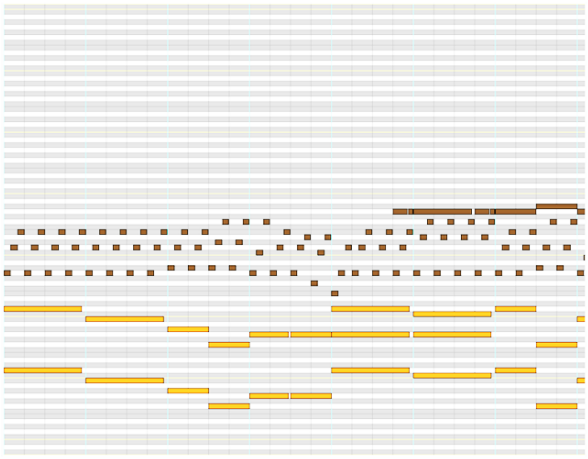
Beethoven Fur Elise



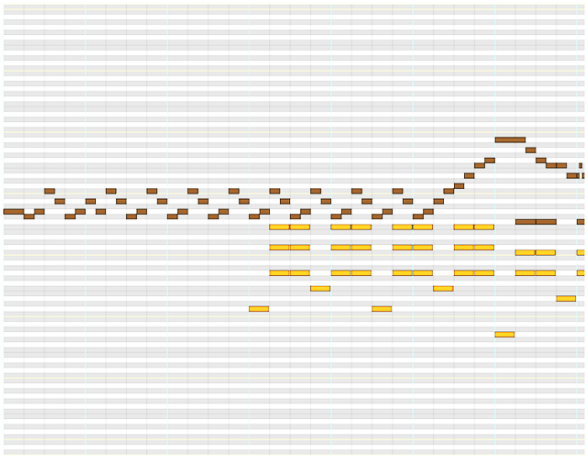
Chopin Trois Valses

Appendix C Examples of different feature recombination

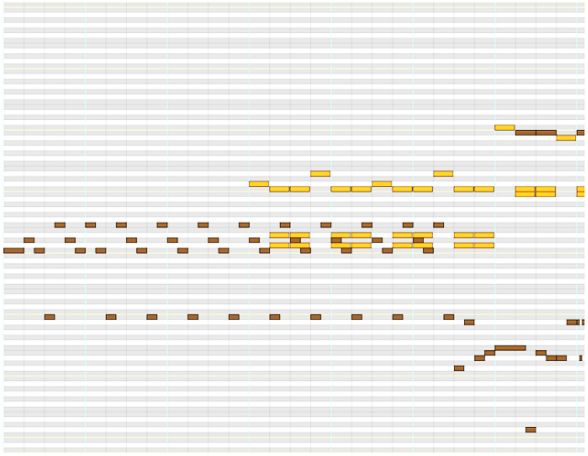
Different results of a rhythmic section of a composition recombine with the harmonic section of a different composition.



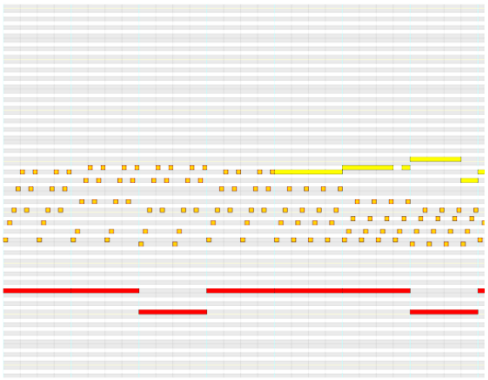
Beethoven: Piano Sonata No. 14 Moonlight



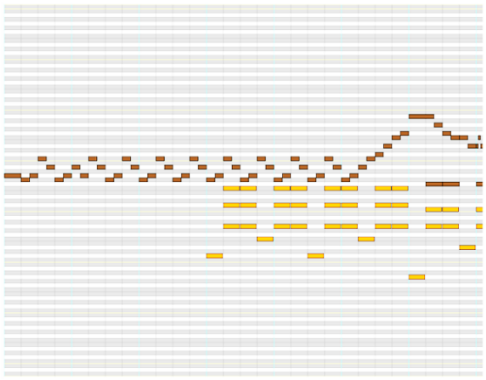
Chopin: Trois Valses



Chopin rhythm with Beethoven harmony

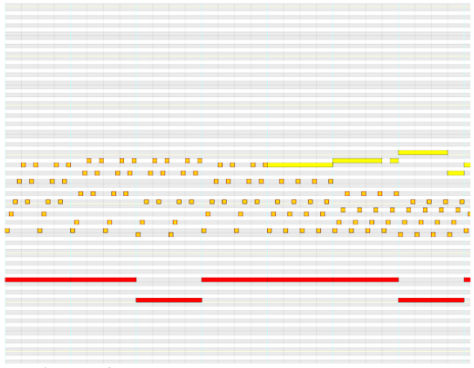


Bach: Ave Maria



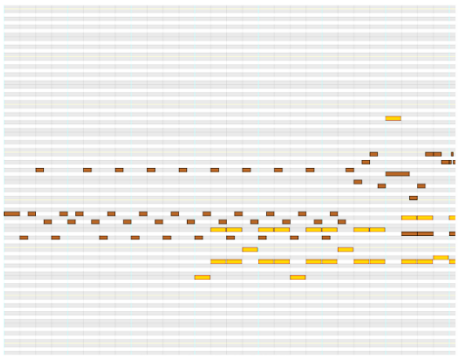
Chopin: Trois Valses

Bach rhythm with Chopin harmony



Bach: Ave Maria

Chopin: Trois Valses



Chopin rhythm with Bach harmony

Appendix D Music repository explanation of examples

These are examples of the final method rendered with virtual instruments of high quality.

These compositions are polyphonic and result in the recombination of the harmonic development of different songs. They preserve the onsets of the rhythmic sections, and the instruments were selected after the composition was finished. Percussion remained the same as the original rhythmic sections.

https://hitrecord.org/users/procopio

In this other repository, the examples were rendered with sound-fonts, which are samples of instruments and sliced to sound as independent sounds with prerecorded behavior. The quality is not as the latest architecture of music composition, but these were the first examples during the methodology exploration. These are also polyphonic, but the recombination was mainly randomized.

https://soundcloud.com/omar-corner