MUSICAL COMPOSITION BY USING GENETIC ALGORITHMS AND ARTIFICIAL NEURAL NETWORKS

by

Umut Karakulak

Engineering Project Report

Yeditepe University

Faculty of Engineering and Architecture

Department of Computer Engineering

2015

MUSICAL COMPOSITION BY USING GENETIC ALGORITHMS AND ARTIFICIAL NEURAL NETWORKS

APPROVED BY:

Assoc.Prof.Dr. Emin Erkan Korkmaz (Supervisor)	………………………………
Assist.Prof.Dr. Dionysis Goularas	………………………………
Prof.Dr. Semih Bilgen	………………………………

DATE OF APPROVAL: /   /2016

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Assoc. Prof Dr. Emin Erkan Korkmaz for his creative guidance and support throughout this project. Also I would like to thank Assist. Prof Dr. Dionysis Goularas for sharing his knowledge in the music domain.

Also I would like to thank Barış Dai, Tuğçe Hakarar and Nurgül Canatan for their patience and help during our music theory lessons as well as their constructive support.

Finally, I would like to thank the volunteers that have participated in my tests and Ahmet Atasoy and Kemal Çağrı Serdaroğlu for supervising the test.

ABSTRACT

MUSICAL COMPOSITION BY USING GENETIC ALGORITHMS AND NEURAL NETWORKS

This project suggests an approach to create musical compositions using artificial intelligence. With the use of stacked Boltzmann machines it aims to learn pitch, duration and velocity values of compositions. This smaller feature set of musical compositions are then clustered to find the average features of the songs by using k-means algorithm.

Project relies on the hypothesis that these average feature set represents the basis of music. Then these averages are used in the evaluation function of a genetic algorithm. The process of building an evaluation function from neural networks removes the possible human bias and creates a composition with little interference.

This way, the program can perform learning on a given dataset and create musical compositions in that given domain; which provides a general framework that can be used to imitate specific musicians by creating a dataset with their songs or to compose in a genre by using a specific genre as the dataset.

This project will be evaluated by a human audience on how well it performs against a random string of notes. The evaluation process will be done with multiple outputs to conclude if the project can extract a proper musical structure from the dataset and generate interesting results.

ÖZET

GENETİK ALGORİTMALAR VE YAPAY SİNİR AĞLARI İLE MÜZİK BESTELERİ

Bu proje yapay zeka kullanılarak, müzik besteleme alanına bir bakış açısı öne sürer. Katmanlı sınırlandırılmış Boltzmann makinaları ile bestelerin perde, uzunluk ve sertlik özelliklerinin öğrenilmesi amaçlanmıştır, elde edilen bu özellik kümesinin daha sonra k-means algoritması kullanılarak ortalamaları alınır.

Proje bu özellik kümesi ortalamalarının müziğin temelini verdiği hipotezine dayanır. Bu ortalamalar daha sonrasında genetik algoritmaların değerlendirme fonksiyonlarında kullanılır. Değerlendirme fonksiyonunu yapay sinir ağları ile oluşturma süreci, program sonucunu değiştirebilecek olası insan etkisini ortadan kaldırarak dış etkenlerden bağımsız bir sonuç yaratmaya yardımcı olur.

Böylece program, girilen veri kümesini öğrenebilir ve besteler yaratabilir. Bu yapı sayesinde programa belirli veri kümeleri yüklenerek programın bir müzik türüne veya bir müzisyenin şarkılarına benzer şarkılar üretmesi sağlanabilir.

Proje’nin testi dinleyiciler tarafından yapılarak rastgele nota dizini karşısındaki performansı değerlendirilecektir. Bu değerlendirme süreci birkaç farklı proje çıktısı ile tekrarlanarak, projenin veri kümesinden düzgün bir müzik yapısı oluşturması ve ilgi çekici müzikler geliştirmesi sonuçları incelenecektir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

Avg	Average.
CD-K	Persistent contrastive divergence learning with k steps of alternate Gibbs sampling.
CNN	Convolutional Neural Network.
E(v,h)	Energy of visible and hidden layers of a restricted Boltzmann machine.
GA	Genetic Algorithms.
h_i	Neuron in a hidden layer.
k	Number of clusters in K-means.
LSTM	Long Short Term Memory Network.
max()	Maximum of a population.
RBM	Restricted Boltzmann Machine.
RNN	Recurrent Neural Network.
sign()	Mathematical signum function, extracts the sign of a numeric value.
v_i	Neuron in a visible layer.
w_i	Weight of an artificial neuron.
WPF	Windows Presentation Foundation.

1. INTRODUCTION

With the constant development and success in machine learning and artificial intelligence, it is inevitable that algorithms will replace humans in some domains. But art has always been a topic of discussion and has been a challenge for enthusiasts, especially after the success of convolutional neural networks.

In computer generated music, genetic algorithms are vastly popular due to their convenient structure for music representation and ability to find close to optimal solutions in large problem spaces. In recent years long short term memory (LSTM) neural networks and pattern recognition techniques are also used in automatic music composition due to their capabilities to imitate human compositions.

1.1. Problem Definition

Although genetic algorithms provide a great way for searching musical solutions, a big problem appears when they are used for the music domain; genetic algorithms rely on finding better fit solutions using a formula called evaluation function or fitness function. In music domain, it is hard to formalize such a criteria.

Previous works include using human feedback as the fitness function where people would vote on their preferences (Brad Johanson, 1998 et al.) [1]. An alternative method is using music theory rules on a seed note string to evaluate the fitness of a complex composition structure (Nicolas Tollervey, 2012) [2]. Creating a fitness function based on human intuition exists in the literature too, as described by Dostál and Martin [3].

Although these can be accepted as solutions to the problem of finding fitness function for computational music generation, their weaknesses can outweigh the solution. Using human feedback in evaluation function requires tens of man-hours for a composition depending on the generation and population sizes, human intuition and feedback may affect the output to be biased by the voters or the programmer’s taste and finally music theory rules are only effective on some subset of this big solution set and may only result strict outputs.

With the recent progress on the LSTM networks on the other hand, neural network solutions may output better result. But due to the nature of these algorithms, these networks act as complex pattern recognition methods and result in orderly copies of previous works. That is lack of this creative spark that genetic algorithms may present.

This project aims to tackle this implicit fitness function problem in genetic algorithms by using a hybrid approach; which utilizes deep belief networks to extract common feature sets of music, and uses clustering to take the mean average of those features.

The key idea of this project is that the mean values collected from the clustering algorithm would give a general structure of music. Thus, cluster results are then used in a reward-penalty fitness function of a genetic algorithm framework. The fitness values of potential solutions generated by the genetic algorithm are determined based on their distance to the closest cluster mean.

1.2. Requirements

Given enough training and the dataset features, the activation vector of a selected neural network should carry the following information about a song;

- The duration of a note.
- The timestamp of a note, extracted explicitly by recursive sum of the notes played earlier.
- Pitch of a note.
- Velocity of a note.

With this information at hand, the algorithm should be able to find an average sounding song by using clustering. This output will be the general fundamental structure to be built upon during the genetic algorithm iterations.

Finally taking this structure as basis, the algorithm should be able to generate creative but structurally relevant to what a human creates as an output in reasonable time.

1.3. Evaluation of Results

Project results will be evaluated by the feedback of human audience. Audience members will judge the project’s capability of extracting this fundamental structure and generation of results upon this structure by listening to results.

Audience members will listen to multiple outputs of the basis achieved during clustering, genetic algorithm results using those bases and random note strings for comparison in no peculiar order.

Tests will be done with uninformed audience; where participants will have zero knowledge about the songs played and rate the outputs according to its quality of composition and how interesting they thought it was. Finally, audience members will be asked to guess the random note string to have a better understanding of project’s outputs.

Test results will be evaluated by taking audience members background on music theory knowledge and their votes on how well they think their musical ear are.

2. BACKGROUND

2.1. Software Background

Project has been developed in .NET framework using C# interoperation with MATLAB. Scrum project management has been adopted and during project's lifespan and Microsoft Team Foundation Server has been used to deploy the code and the weekly schedule.

2.1.1. Restricted Boltzmann Machines

RBMs are energy based stochastic networks that consist of one layer of visible neurons and one layer of hidden neurons. RBMs are symmetric bipartite graphs, meaning each visible neuron is connected to each hidden neuron through a symmetric bi-directional weight and each hidden neuron is connected to each visible neuron through a symmetric bi-directional weight. And restriction means no visible unit is connected to another visible unit and no hidden unit is connected to another visible unit.

Activation of the nodes is based on a probabilistic weighted sum of all nodes connected to it. A graphical representation of the nodes can be seen in the Figure 2-1 below, where grey nodes represents the hidden layers and white nodes represents the visible layers.

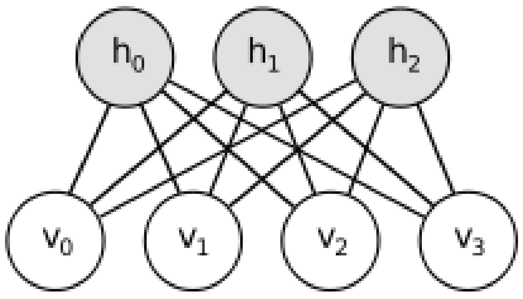


Figure 2-1 Restricted Boltzmann machine.

Energy function between hidden and visible units of a standard type restricted Boltzmann machine is given in Equation 2-1.

$$E(v,h) = -\sum_{i\in visible} a_i v_i - \sum_{j\in hidden} b_j h_j - \sum_{ij} v_i h_j w_{ij} \qquad (Equation\ 2-1)$$

By feeding an RBM's visible layer output to another RBM's visible layer, stacks can be formed to create a deep belief network as proposed by Hinton et al. 2006 [4]. It has been concluded that the amount of hidden layers increases the complexity of features extracted from the dataset.

2.1.2. Contrastive Divergence Learning

The idea of this unsupervised learning method has been first put up by Geoffrey E. Hinton in 2002[5] as a way to speed up the training process and has proven to achieve similar results to probabilistic learning methods.

Initializations for the weight values are done using a Gaussian field projection. Algorithm starts with a forward pass that activates the hidden neurons probabilistically. These output probabilities are then sampled and used to activate visible neurons using the symmetry of RBMs. This forward and backward pass is called alternating Gibbs sampling, a representation can be seen in Figure 2-2 below.

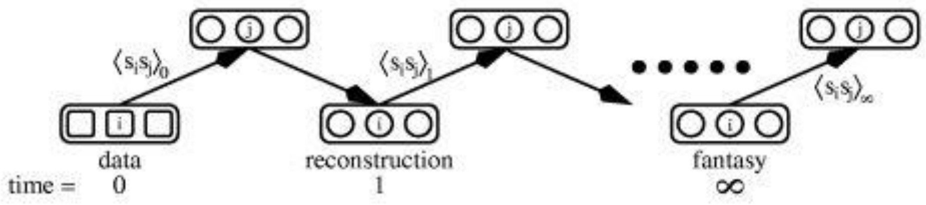

$\langle s_i s_j \rangle_0$
$\langle s_i s_j \rangle_1$
$\langle s_i s_j \rangle_\infty$
data
reconstruction
fantasy
time =
0
1
∞

Figure 2-2 Visualization of alternating Gibbs sampling, fantasy and reconstruction.

Algorithm then activates the hidden neurons once more with this reconstruction sample and calculates error on how distant the fantasy vector was from the original input. A pseudocode[6] written as the description in Hinton's paper and used to implement contrastive divergence learning in this project can be found in Table 2-1.

Table 2-1 Pseudocode for CD-K (contrastive divergence) learning algorithm.

Algorithm 1 Contrastive divergence

```
function contrastiveDivergenceLearning
    inputs: $x_1$, $\epsilon$, W, b, c
    $x_1$ is a sample from the training distribution for the RBM
    $\epsilon$ is a learning rate for the stochastic gradient descent
    W is the RBM weight matrix, of dimension (hidden units, inputs)
    b is the RBM offset vector for input units
    c is the RBM offset vector for hidden units
    for all hidden units i do
        compute $Q(h_{1i} = 1|x_1)$ (for binomial units, $\mathrm{sigm}(c_i + \sum_j W_{ij}x_{1j}))$
        sample $h_{1i} \in \{0,1\}$ from $Q(h_{1i}|x_1)$
    endfor
    for all visible units j do
        compute $P(x_{2j} = 1|h_1)$ (for binomial units, $\mathrm{sigm}(bj + \sum_i W_{ij}h_{1i}))$
        sample $x_{2j} \in \{0,1\}$ from $P(x_{2j} = 1|h_1)$
    endfor
    for all hidden units i do
        compute $Q(h_{2i} = 1|x_2)$ (for binomial units, $\mathrm{sigm}(c_i + \sum_j W_{ij}x_{2j}))$
    endfor
    $W \leftarrow W + (h_1\acute{x}_1 - Q(h_{2.} = 1|x_2)\acute{x}_2)$
    $b \leftarrow b + \epsilon(x_1 - x_2)$
    $c \leftarrow c + \epsilon(h_1 - Q(h_{2.} = 1|x_2))$
endfunction
```

2.1.3. K-Means Clustering

K-means is a method used to partition multidimensional data into k predetermined number of clusters. Algorithm works on Euclidean space and assigns the multivariate vectors using sum of pairwise square of Euclidean distances.

At the first run it finds the most distant k individuals and determines them as k clusters with centroids as their mean vector. Then every remaining individual is introduced to a group with closest Euclidean distance. Once all of the individuals have been assigned to a cluster, centroids are recalculated. The last two steps are repeated until there is no change in centroid vectors or a certain criteria is met.

2.1.4. Genetic Algorithms

Genetic algorithms, inspired by Darwinist principle of survival of the fittest are a metaheuristic framework that is used to solve large and ill-behaved solution spaces.

Solutions are encoded into a string of symbols, that is usually in binary are called chromosome. These solutions are then evolved into better fit solutions using crossover and mutation operators throughout generations until a certain criteria is met. The basic framework consists of the following components;

- Initialization: Population of solutions are generated randomly, using a heuristic from problem domain or based on information from earlier simulations.
- Fitness Evaluation: Individuals in population are rated according to how close they are to the solution; this function decides which individual is better fit.
- Selection: Individuals are selected to mate, randomly or based on their fitness. Some of the most commonly used selection schemes are tournament selection, roulette selection etc.
- Crossover: Creates a new offspring consisting of two parent’s genes, exploitation part of the metaheuristic.
- Mutation: Mutates a solution to expand the search space, exploration part of the metaheuristic.
- Replacement: Replaces the old generation with the selected offspring.

2.2. Previous Works

2.2.1. GenJam: A Genetic Algorithm for Generating Jazz Solos

GenJam [7] is a genetic algorithm based real time jazz music improviser developed in Macintosh/Think C environment and follows generic genetic algorithm structure with custom crossover and mutation operators for music domain. GenJam prefers to start the initial population with a seed value given from the user for play-along feature. General framework on how GenJam works can be seen in Figure 2-3.

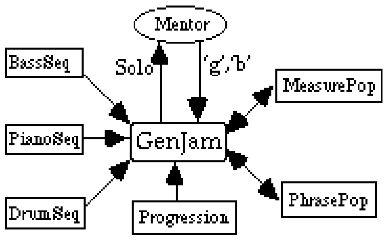


Figure 2-3 GenJam system architecture.

To improvise on a tune, GenJam reads a progression file, the number of solo choruses it should take, and the chord progression along with the pre-generated MIDI sequences for piano, bass and drums.

Fitness values for each solution set are then calculated using a mentor. Mentor is a pre-created database of solo's generated by GenJam and human feedback to those solos.

2.2.2. Learning to Create Jazz Melodies Using Deep Belief Nets

This paper [8] proposes the use of a deep belief network constructed by restricted Boltzmann machines to learn and recreate jazz chords. The training set has been divided into beats and then subdivisions called "slots" before inserted into the first hidden layer.

Then an unsupervised training method of contrastive divergence learning is used to train the deep belief network. To ensure the transition between chord progressions, multiple activations of the network's visible layer are clumped up together in a continuous string as can be seen in the Figure 2-4.

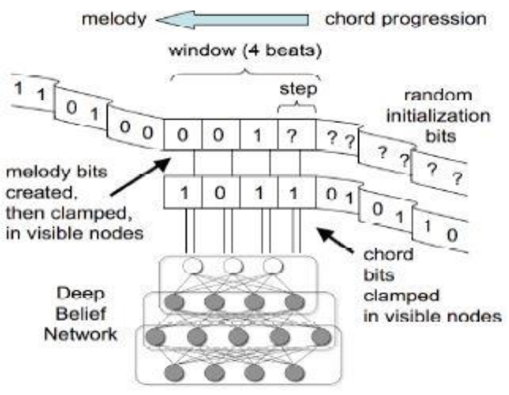

melody
chord progression
window (4 beats)
step
random initialization bits
melody bits created, then clamped, in visible nodes
chord bits clamped in visible nodes
Deep Belief Network

Figure 2-4 An illustration of the process of windowed generation.

To get the results, network is initialized with a seed of chord bits for desired chord progression and random melody inputs and resulting progression bits are compiled as explained above from the hidden layer of last RBM.

2.2.3. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription

This paper [9] tackles the problem based on an idea with restricted Boltzmann machines that can hold a memory of the parameters from a previous iteration called temporal RBM that is put up by Ilya Sutskever, Geoffrey E. Hinton and Graham W. Taylor [10]. And introduces a generalization of that model called RNN-RBM, which allows more freedom.

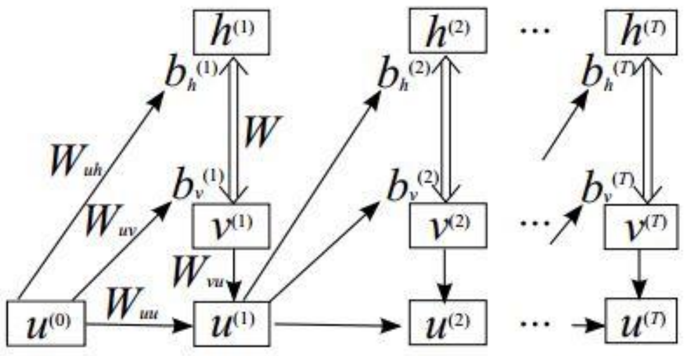


Figure 2-5 RNN – RBM.

The RNN-RBM structure can be simplified as a sequence of conditional RBMs whose input parameters are the output of a RNN. In the Figure 2-5, upper half represents the RBM stage while the lower half represents the RNN with hidden units.

RNN part of the structure handles the time dependency and produces a set of outputs that are then used as the parameters for the RBM. RBM then models the conditional distribution and decides on which notes should be played in probabilistic relation.

3. ANALYSIS & DESIGN

This project aims to generate songs by using genetic algorithms with a fitness function that tries to find a solution that is closest to an average basis representation of songs. This basis is achieved by extracting important features from a set of songs using a neural network, composed of stacked restricted Boltzmann machines, outputs of this network are than clustered to find the average basis.

Thus, project should be analyzed and designed considering three main stages as shown with a simplified flowchart in Figure 3-1 below;

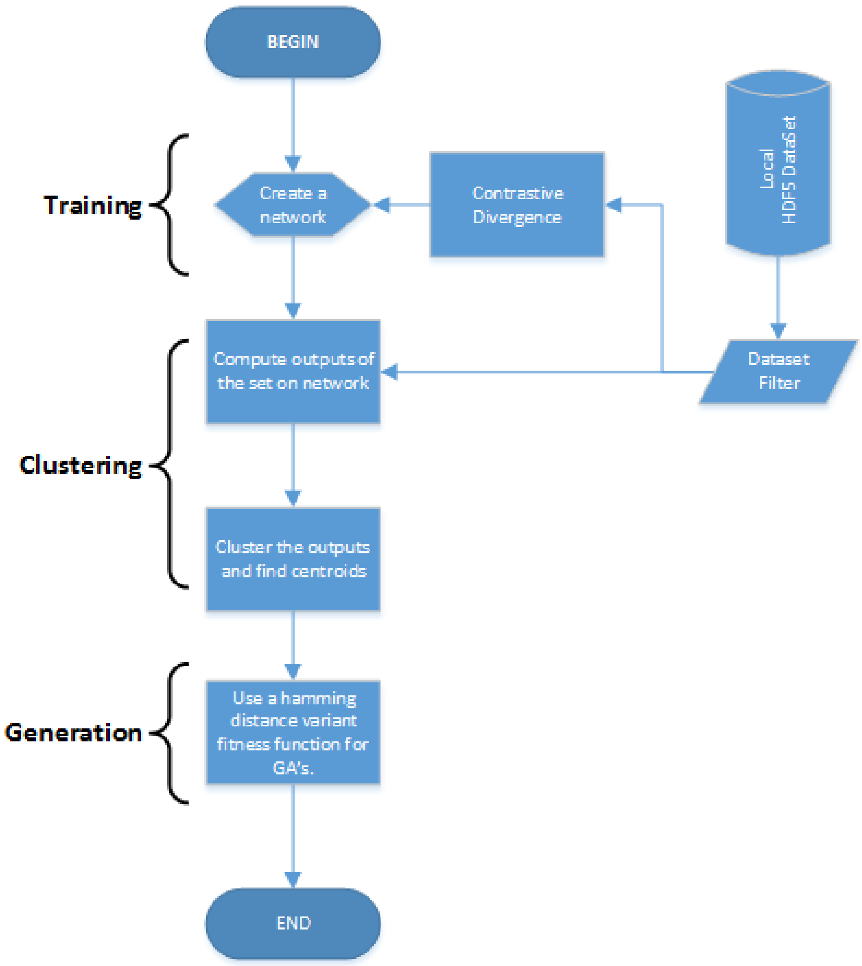

BEGIN
Training
Create a network
Contrastive Divergence
Local HDF5 DataSet
Compute outputs of the set on network
Dataset Filter
Clustering
Cluster the outputs and find centroids
Generation
Use a hamming distance variant fitness function for GA's.
END

Figure 3-1 General framework of the project.

- Training: Desired song features are filtered and extracted using neural networks.
- Clustering: Feature vectors are clustered to find average features of the dataset. Output should be the feature set of a dull but acceptable song.
- Generation: Using the averages acquired in clustering stage as a building block of fitness function, generates a new song.

3.1. Modelling Music

In genetic algorithms, solutions are represented as strings of symbol sets, usually in binary form. For consistency model that is designed in this step will be either used on all components or needs to be easily converted, thus bears great importance.

I have designed a simple 16 bit musical note genome for performance, so that a 32 bit architecture cache can carry two notes at a time and a genome can still carry enough information. Model genome uses the same labels as the dataset for consistency, which is displayed in Figure 3-2. Octave information is left out due to dataset insufficiency.

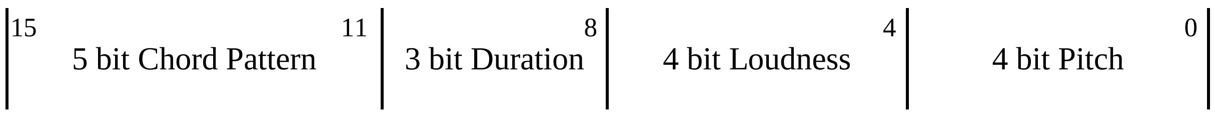


Figure 3-2 Sections of genome string.

Pitch values can range from C to B including semitones, with the silent note there are 13 possible pitch values.

Possible loudness values are reduced to ease on the complexity of genetic algorithms and neural networks due to their little effect on the outcome. Loudness value in model affects the midi output by multiplication of five and has a negative effect, so more loudness in model means less velocity on output.

For duration, basic timestamps in music theory was taken as displayed in Figure 3-3;

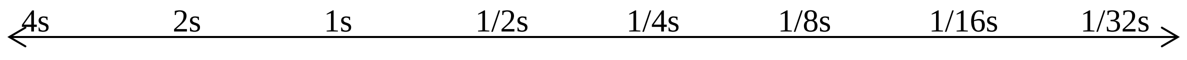


Figure 3-3 Range of duration values.

Model can be in two form, single note or chord note. So far these three features satisfy single note form. If the genome’s 11-15 bits are not empty genome is a chord note and chord pattern data also needs to be read. Chord pattern holds an array of bytes, which describes; how far the notes are from the root note, root note being pitch in this case. Supported chords can be seen in Table 3-1.

Table 3-1 Chord patterns recognized by model.

Chord Name	Intervals of Chords
Major	0,4,3
Minor	0,3,4
Fifth	0,7
Dominant7th	0,4,3,3
Major7th	0,4,3,4
Minor7th	0,3,4,3
MinorMajor7th	0,3,4,4
Sus4	0,5,2
Sus2	0,2,5
Sixth	0,4,3,2
Minor6	0,3,4,2
Ninth	0,2,2,3,3
Minor9	0,2,1,4,3
Major9	0,2,2,3,4
MinorMajor9	0,2,1,4,4
Eleventh	0,2,2,1,2,3
Minor11	0,2,1,2,2,3
Major11	0,2,2,1,2,4
MinorMajor11	0,2,1,2,2,4
Thirteenth	0,2,2,3,2,1
Minor13	0,2,1,4,2,1
Major13	0,2,2,3,2,2
MinorMajor13	0,2,1,4,2,2
Add9	0,2,2,3
MinorAdd9	0,2,1,4
SixAddNine	0,2,2,3,2
Minor6Add9	0,2,1,4,2
Dominant7thAdd11	0,4,1,2,3
Major7thAdd11	0,4,1,2,4
Minor7thAdd11	0,3,2,2,3
MinorMajor7thAdd11	0,3,2,2,4
Dominant7thAdd13	0,4,3,2,1

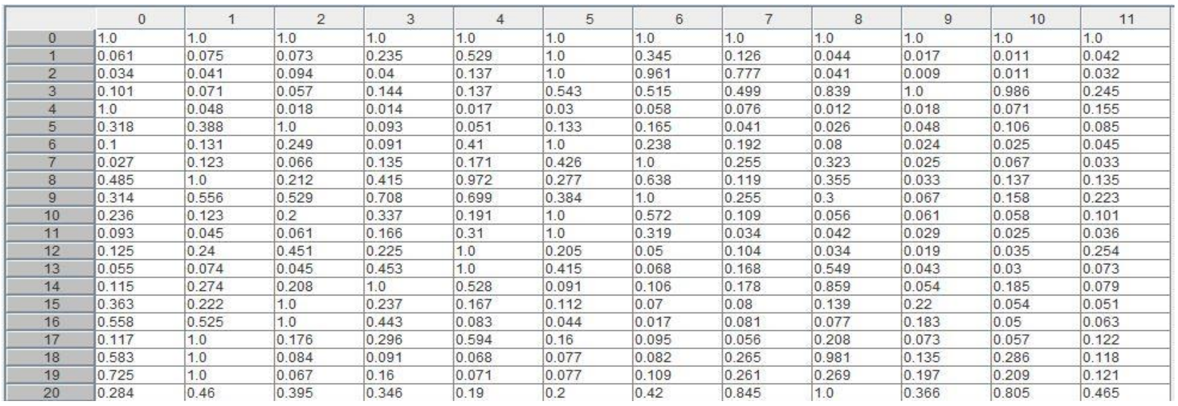
3.2. Dataset Collection

I’ve used the million song dataset [11], a collection of audio features and metadata of songs in HDF5 format. Since loading the dataset is a one time job and HDF5 is a O(1) complexity library, optimization was not a concern.

I’ve focused on the data integrity instead. I have downloaded a random fraction of the 280 GB dataset available online. Features were fuzzy estimations, thus I have decided to use “rock” genre and similar tempo songs and have experimented with different threshold values to load songs to my program.

The thresholds were used in two features of dataset, genre and pitch, which values were differing between 1 and 0. After doing some experiments I’ve decided to use 0.9 for genre, which have satisfied me for 500 songs.

For pitch values I’ve selected the maximum value of a row for each note in the matrix representation of pitch values that can be seen in the Figure 3-4 below. Columns in the figure represent the pitch value of a note from C to B including semitones, while the rows represent the notes of a song in order of their playback.



	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	0.061	0.075	0.073	0.235	0.529	1.0	0.345	0.126	0.044	0.017	0.011	0.042
2	0.034	0.041	0.094	0.04	0.137	1.0	0.961	0.777	0.041	0.009	0.011	0.032
3	0.101	0.071	0.057	0.144	0.137	0.543	0.515	0.499	0.839	1.0	0.986	0.245
4	1.0	0.048	0.018	0.014	0.017	0.03	0.058	0.076	0.012	0.018	0.071	0.155
5	0.318	0.388	1.0	0.093	0.051	0.133	0.165	0.041	0.026	0.048	0.106	0.085
6	0.1	0.131	0.249	0.091	0.41	1.0	0.238	0.192	0.08	0.024	0.025	0.045
7	0.027	0.123	0.066	0.135	0.171	0.426	1.0	0.255	0.323	0.025	0.067	0.033
8	0.485	1.0	0.212	0.415	0.972	0.277	0.638	0.119	0.355	0.033	0.137	0.135
9	0.314	0.556	0.529	0.708	0.699	0.384	1.0	0.255	0.3	0.067	0.158	0.223
10	0.236	0.123	0.2	0.337	0.191	1.0	0.572	0.109	0.056	0.061	0.058	0.101
11	0.093	0.045	0.061	0.166	0.31	1.0	0.319	0.034	0.042	0.029	0.025	0.036
12	0.125	0.24	0.451	0.225	1.0	0.205	0.05	0.104	0.034	0.019	0.035	0.254
13	0.055	0.074	0.045	0.453	1.0	0.415	0.068	0.168	0.549	0.043	0.03	0.073
14	0.115	0.274	0.208	1.0	0.528	0.091	0.106	0.178	0.859	0.054	0.185	0.079
15	0.363	0.222	1.0	0.237	0.167	0.112	0.07	0.08	0.139	0.22	0.054	0.051
16	0.558	0.525	1.0	0.443	0.083	0.044	0.017	0.081	0.077	0.183	0.05	0.063
17	0.117	1.0	0.176	0.296	0.594	0.16	0.095	0.056	0.208	0.073	0.057	0.122
18	0.583	1.0	0.084	0.091	0.068	0.077	0.082	0.265	0.981	0.135	0.286	0.118
19	0.725	1.0	0.067	0.16	0.071	0.077	0.109	0.261	0.269	0.197	0.209	0.121
20	0.284	0.46	0.395	0.346	0.19	0.2	0.42	0.845	1.0	0.366	0.805	0.465

Figure 3-4 Pitch values of a song from million song dataset.

I have then declared that as my root pitch and if any other value satisfies the Equation 3-1 below, added that value to a list as a possible part of a chord in that root.

$$(Root \leq Pitch_i + Threshold) \qquad (Equation\ 3-1)$$

3.3. Stacked Boltzmann Machines

For the training part of the project I have discussed different neural network implementations and experimented with different sizes of networks and layers.

Auto encoders were initially considered to be the neural networks used in this project but for the similarity in the encoding part and simpler nature, restricted Boltzmann machines were decided to be used.

Three belief networks are designed for three features instead of one big network, due to the length of the input layer and better performance considerations. The first network utilized is the segment network that is responsible for duration of notes. The second one is the pitch network which represents the pitch of the note from C to B and lastly the loudness network is utilized for the velocity of a note when played. Contrastive divergence learning is decided to be used to train these networks for the simplicity and performance.

While designing the segment network, a simple equation as shown below is used to determine the input layer size, where songLength is the length (in seconds) of the data that will be trained and precision is how precise my model can represent a second. For the project I have decided to train the first 60 seconds of all songs and as of my model design my precision is 1/32 of a second, thus segment network's input layer size was decided to be 1920 using the Equation 3-2 below.

$$\left(LayerSize = songLength \times \frac{1}{precision}\right) \qquad (Equation\ 3-2)$$

Segments in dataset was in form of timestamps of notes, hence the matching input neuron is activated for every valid segment value in a song. Similarly, for every note a pitch and loudness value exists, maximum number of notes exists in first 60 seconds of all songs used in segment network training was taken as a base (370) and multiplied by 12 for note and loudness ranges, resulting in 4440.

$$(LayerSize = \max(\#notes\ in\ songs)\ x\ 12) \qquad (Equation\ 3-3)$$

After separating the dataset into 3 networks, inputs were simplified enough to get fine results using a single hidden layer. For calculating the optimal size of this hidden layer; I have done multiple tests and compared the error rates of the different sized networks, starting with the rule of thumb methods for multi-layer neural networks such as using the average of the input and output layer. As a final result, following networks with sizes shown in Figure 3-5 is decided to be used, where segment network structure can be seen on the left and pitch, loudness structures can be seen on the right. Labels “v” in figure represents visible layer and “h” represents hidden layers, while “h2” hidden layer acts as the output of the belief network.

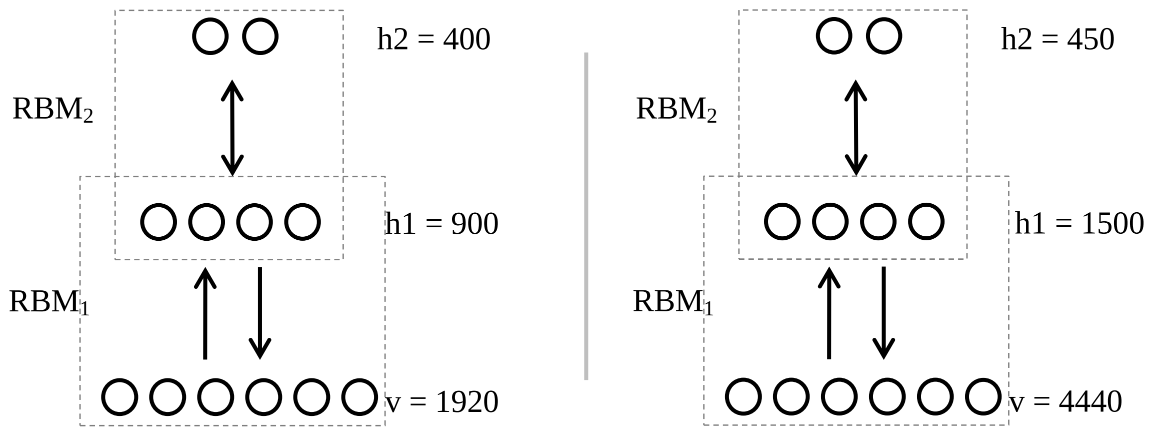


Figure 3-5 Belief network structures for segment, loudness and pitch.

Training was limited with the first 60 seconds of the songs. Due to the nature of pitch values ranging from 0 to 12, input layer was set to 4440. For convenience, loudness range of 60 in dataset has been reduced to 12 and same network has been used. As seen in Figure 3-5, by using the neural networks, it has been possible to reduce the 1920 feature set for each song, to a dimension of 400 for segment information and reduce the 4440 feature set for each song to 450 for pitch and loudness information. Hence, a feature extraction process has been achieved on the raw data by using stacked Restricted Boltzman machines.

3.4. Selecting the k Value for K-Means

Selecting an optimum k value in K-Means algorithm is an important part of the clustering algorithm. To find the optimum value, one can try the algorithm with different values of k multiple times and analyze the silhouette values of the resulting clusters. But this would neither be a good solution nor a fast one.

There are multiple solutions proposed in this area and Matlab has multiple external libraries that can output the indices to compare the quality of clusters. I have used Tool for estimating the number of clusters library [12] which had given me k=20 as where the total distance would be smallest between cluster observations and means. But due to the small number of songs that I have trained 20 clusters are too much; Thus I have decided to use 10 as the number of clusters for my segment network after analyzing silhouette values and calculating total cluster observation and mean distances for k<20 and 5 clusters each for the pitch and loudness networks -if they can satisfy- inside these segment clusters, for a total of 50. Cluster silhouette values can be seen in the Figure 3-6.

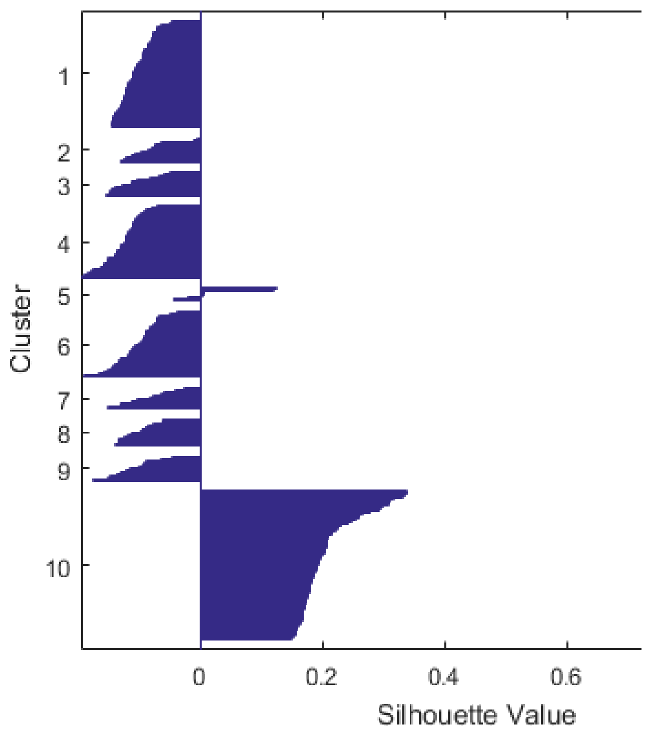


Figure 3-6 Silhouette graph for segment network during k-means with 10 clusters.

Silhouette information is used to determine if a segment cluster was large enough to host pitch and loudness clusters inside, while also analyzing observation distributions. Silhouette values in Figure 3-6 for 10 segment clusters shows that my clusters were sparse enough to give satisfying results when used in fitness function. But 4th and 5th clusters seems too small to have 5 loudness and pitch clusters, while 10th cluster was alienated from the rest of the data and may be dominating, looking at its width. This can cause crowding during generation stage, which can be fixed by using a larger mutation parameter.

Since clustering is a crucial part of my project I wanted to analyze this as best as I can and ran through a hierarchical clustering using Euclidean distance to confirm if the trained set is really fit for clustering, which can be seen in the Figure 3-7.

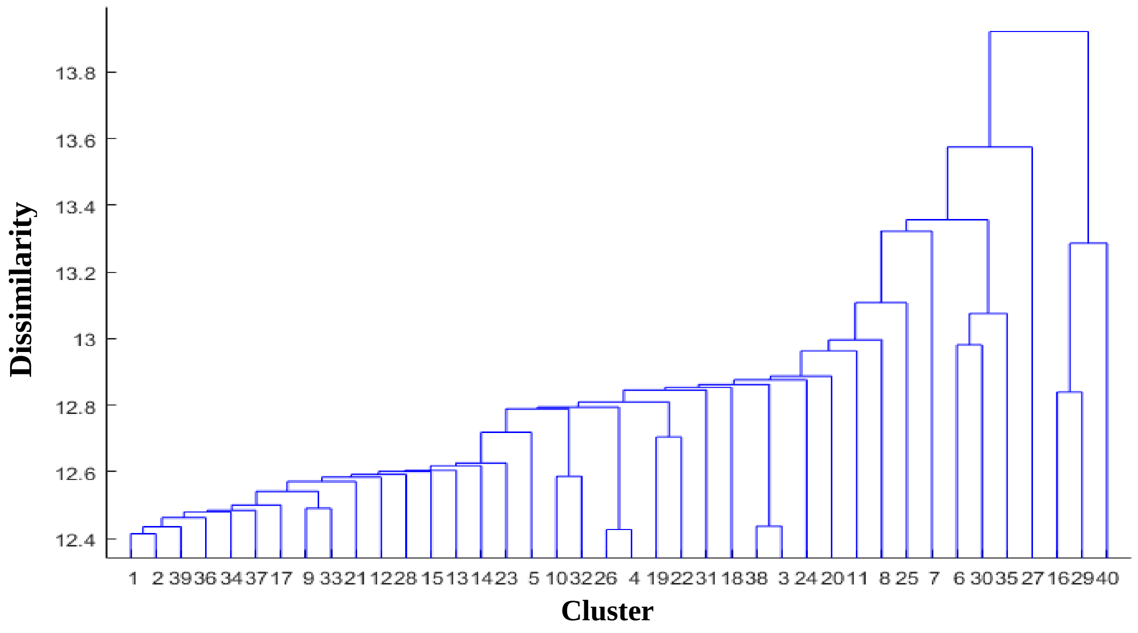


Figure 3-7 Dendogram of Euclidean distances in segments, using hierarchical clustering.

The Figure 3-7 is a dendogram plot for 40 clusters. Splitting of a vertical line shows a cluster while the distance between two splits gives the dissimilarity of two clusters. Looking at the positions of splits in the graph from top to bottom, around 10 clusters seems to be ideal before cluster means start to look too clumped together.

For a final control by hand, I've decided to see if the clustering was successful using a projection to 3D space. For this, I've applied Sammon's projection from Matlab Toolbox for Dimensionality Reduction [13] to k-means result. 2D projection for 10 segment clusters using Sammon’s mapping can be seen in Figure 3-8, where the observations are colored according to their clusters and different colors represents different segment clusters.

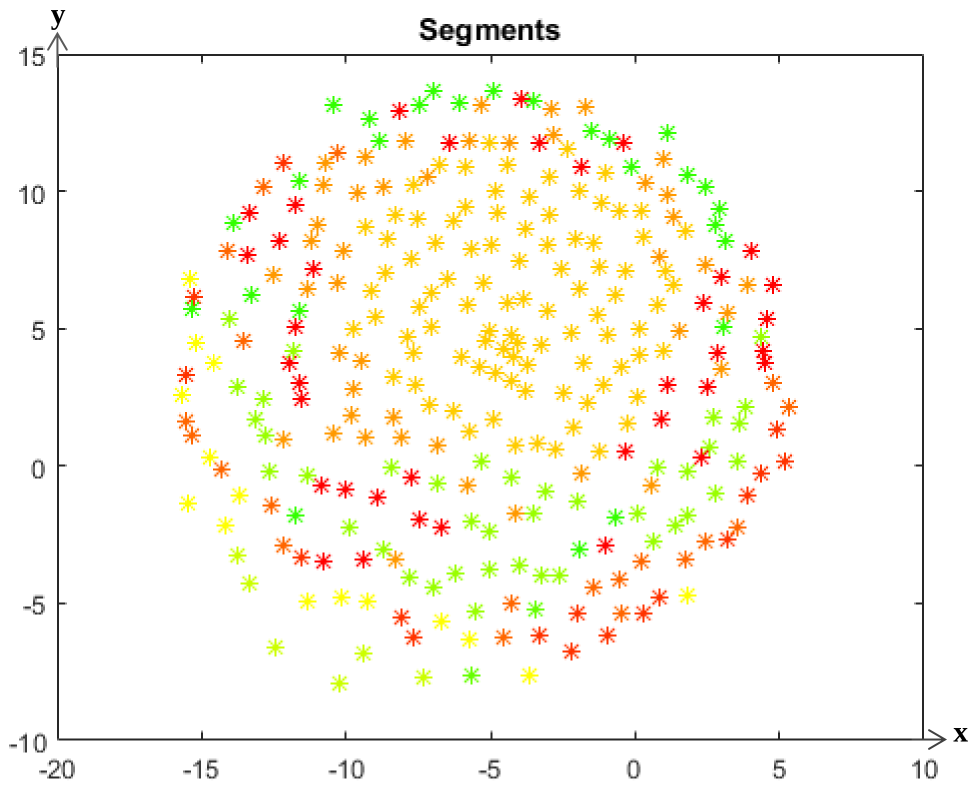


Figure 3-8 Sammon's projection to 2D for segment clustering.

The 2D projection of the segment network outputs range from -20 to 5 in x dimension and -10 to 15 in y dimension. The projection should be viewed as looking at an outer layer of a sphere. When checked in 3 dimensions, different clusters seem to be grouped up in different layers of the said sphere as expected.

Looking at the figure, we can finally conclude that, k-means algorithm was indeed successful and 10 clusters seem to be the most satisfying k number for the segment network outputs.

3.5. Genetic Algorithms

To utilize the information derived in the clustering phase for composing new music, genetic algorithm framework have been developed and the model has been tested by using dummy fitness functions before training implementation, where chromosomes are judged by a combination of human intuition and music theory.

To have a better and faster convergence, tuning parameters for; crossover rate, mutation rate, mutation decline rate and elitism rate is implemented into the metaheuristic framework. These tuning parameters are explained in the following list, values in parentheses show the possible range of values for that parameter.

- Crossover rate (1-0): Chance of the selected couple to have an offspring.
- Mutation rate (1-0): Chance of the selected individual to mutate. Applied to whole genome.
- Mutation decline rate (1-0): Reduction parameter for the mutation rate after every iteration, 1 means no decline.
- Elitism (1-0): Percentage of the best solutions that are carried over the next generation automatically.

After doing multiple convergence tests using the same population size (200), maximum iteration count (750) and seed value for comparison. I’ve decided that although these values are problem dependent and optimal configuration may change once fitness function changes, mutation rate and crossover rate still carries out important decisions and should be set to high values and decline rate didn’t do a lot better in my tests. Thus, I have used 0.75 for crossover rate and 0.2 for mutation rate throughout my tests while tinkering with the other values.

I have decided that parameters like population size and maximum iteration count are problem specific and will be decided upon implementation of proper fitness function.

4. IMPLEMENTATION

Project is built as a desktop application using Windows Presentation Foundation (WPF) using C#. Project can run on .NET framework 2.0 and higher but version 4.0 or higher is necessary for multithreaded training and generation.

With the information acquired from the analysis and design, project is implemented using the same training, clustering and generation structure.

4.1. Class Diagrams

Class diagrams for three main parts of the C# implementation of the program can be seen in the following figures for better understanding on the development process and the evaluation. Diagrams are created in Microsoft .NET standards.

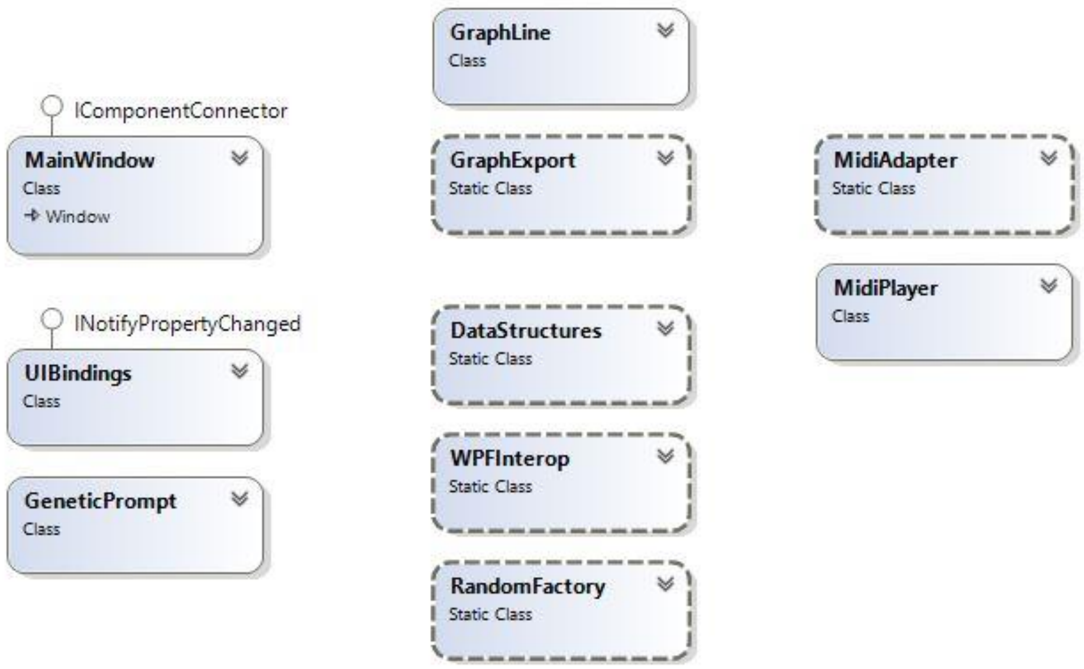


Figure 4-1 UI and midi related classes.

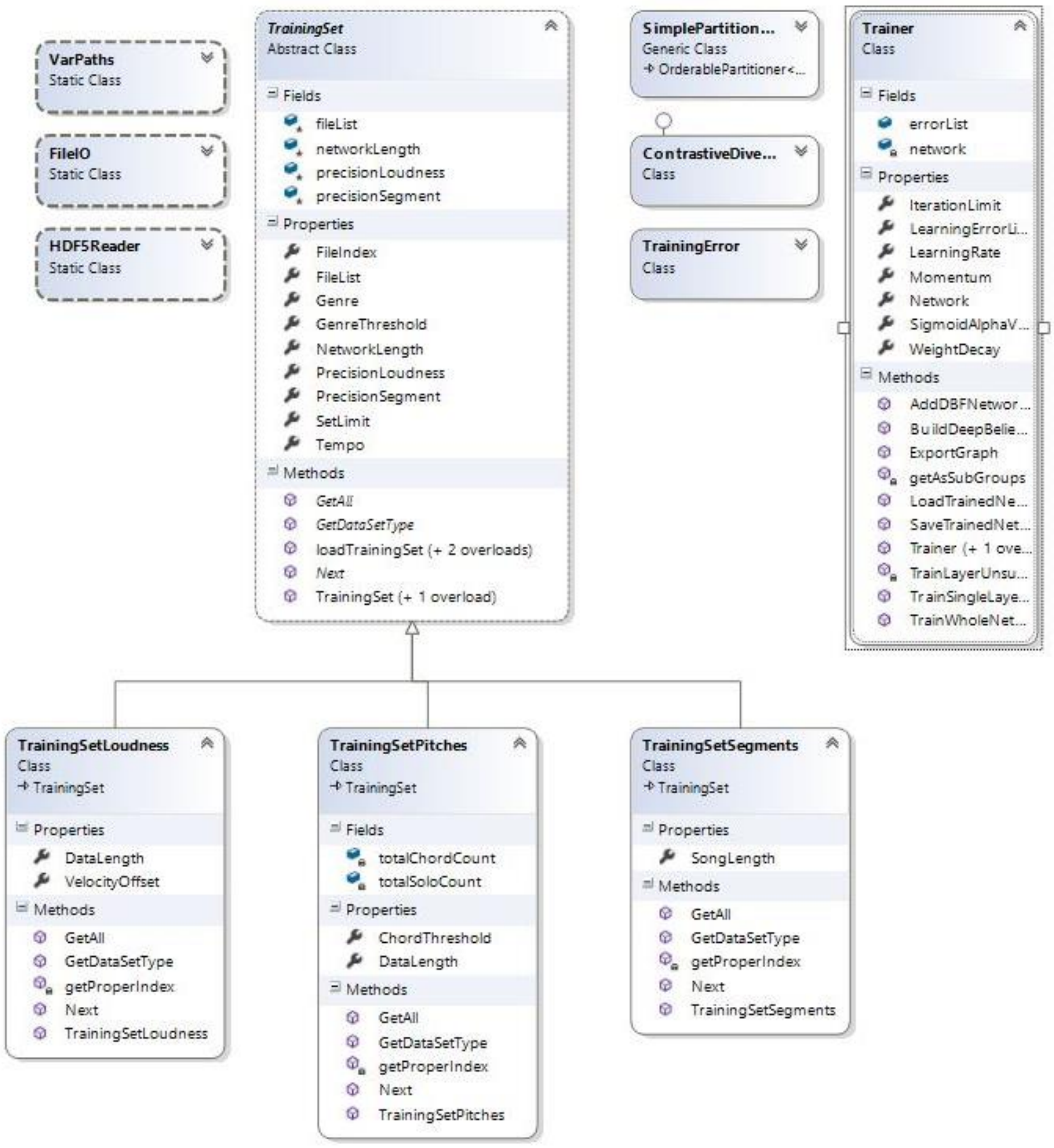


Figure 4-2 Training set and neural network training classes.

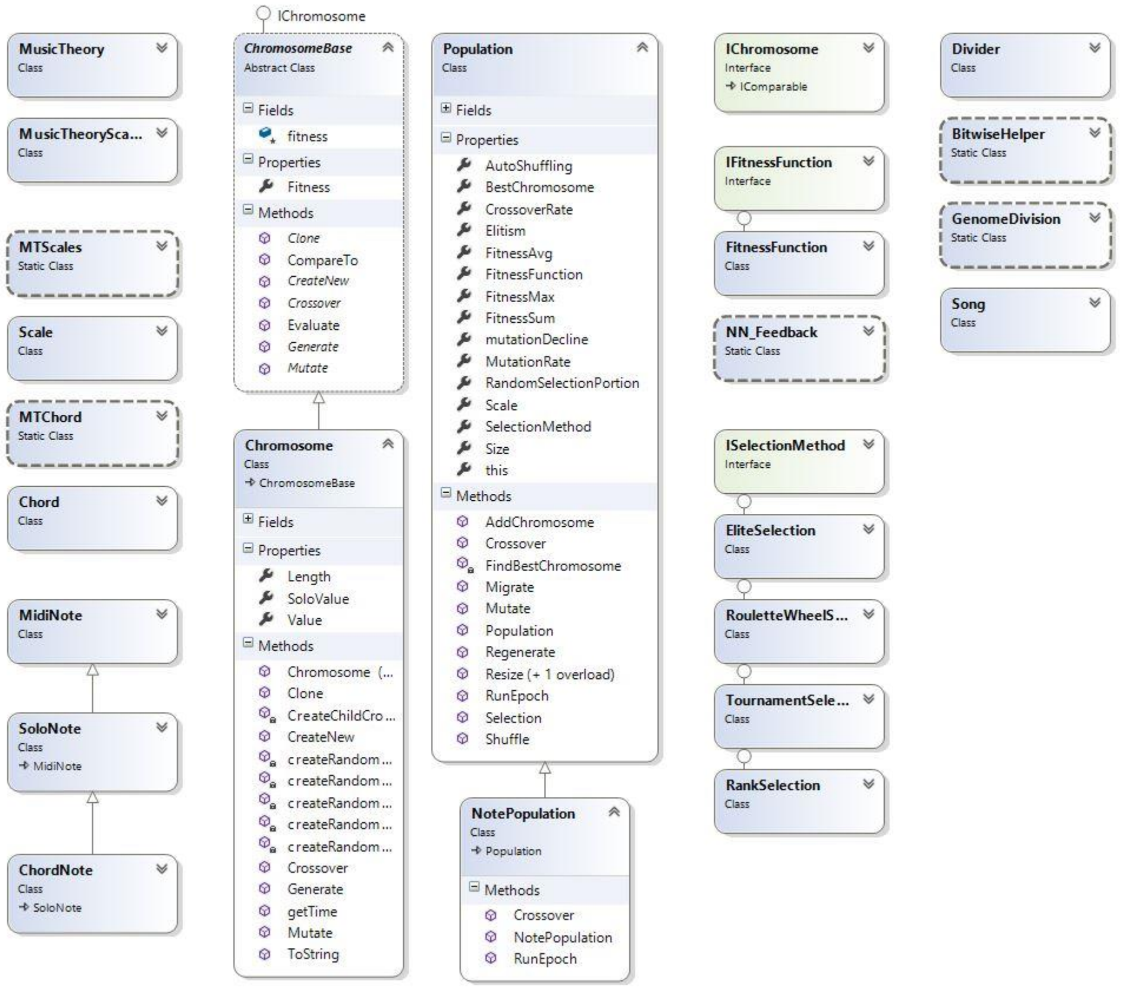


Figure 4-3 Genetic algorithm and song model's classes.

4.2. Training

Accord.NET machine learning framework library [14] with small adjustments on learning classes has been used to perform neural network operations.

A deep belief network object with the designed layer sizes is created and used as a base for stacked Boltzmann machines. To fasten up the process some networks have been serialized and carried over to a cloud based server.

4.2.1. Contrastive Divergence Learning

Greedy layer-wise training was done using contrastive divergence learning method. Training has continued until a certain termination criterion has met.

Training has been done over same 300 instances of rock songs that are randomly selected from the one million song dataset. Learning rate, momentum and weight decay parameters are set close to standards but slightly different for networks by trial and error. A pseudo code for the training can be seen on the Table 4-1.

Table 4-1 Pseudocode for training networks.

Algorithm 2 Pseudocode for Deep Belief Network training.

```
1.  function buildDeepBeliefNet(data)
2.      inputs: data, instance of TrainingSet class.
3.      layerSize ← data.SegmentLength * data.Precision;
4.      this.network = new DeepBeliefNetwork( BernoulliFunction(), layerParams);
5.      this.network.VisibleWeights ←GaussianWeights.Randomize( );
6.      learning ← new DeepBeliefNetworkLearning ( this.Network );
7.      learning.Algorithm ← new ContrastiveDivergenceLearning( );
8.      {
9.          learningRate ← this.LearningRate,
10.         momentum ← this.Momentum,
11.         decay ← this.WeightDecay
12.     };
13.     input = data.GetAll( );
14.     for j ← 0 to (network.layerCount) do
15.         learning.LayerIndex ← j;
16.         layerInput ← learning.GetLayerInput( input );
17.         layerErrors ← Ø;
18.         for i ←0 to (iterationLimit) do
19.             error ← learning.RunEpoch( layerInput );
20.             layerErrors.Add( error );
21.             if (error < 1) then break;
22.         endfor
23.         network.UpdateVisibleWeights( );
24.         networkErrors.Add( layerErrors  );
25.     endfor
26.     this.network.Save( );
27. endFunction
```

Termination criterion was; the iterations would continue until there was no significant gain over error rate using the Equation 4-1 below.

$$Terminate\ ? = Sign\left(error_i - Avg\left(\sum_{i-51}^{i-1} error_i\right)\right) \qquad (Equation\ 4-1)$$

Training has been done using 8 threads in an i7 processor laptop for maximum performance and statistics are given in the tables 4-2 and 4-3.

Table 4-2 Number of iterations over networks.

Machine/Iteration	Segment	Pitch	Loudness
RBM_1	1000	400	400
RBM_2	1000	4000	8000

Table 4-3 Error rates for networks.

Machine/Error Rate	Segment	Pitch	Loudness
RBM_1	0,667046	1,082697	0,882623
RBM_2	5,153865	6,110037	3,911820

Error rate during first 600 iterations of segment network’s training can be seen in Figure 4-4, where blue line represents the error rate of first RBM over time and red line represents the error rate of second RBM over time.

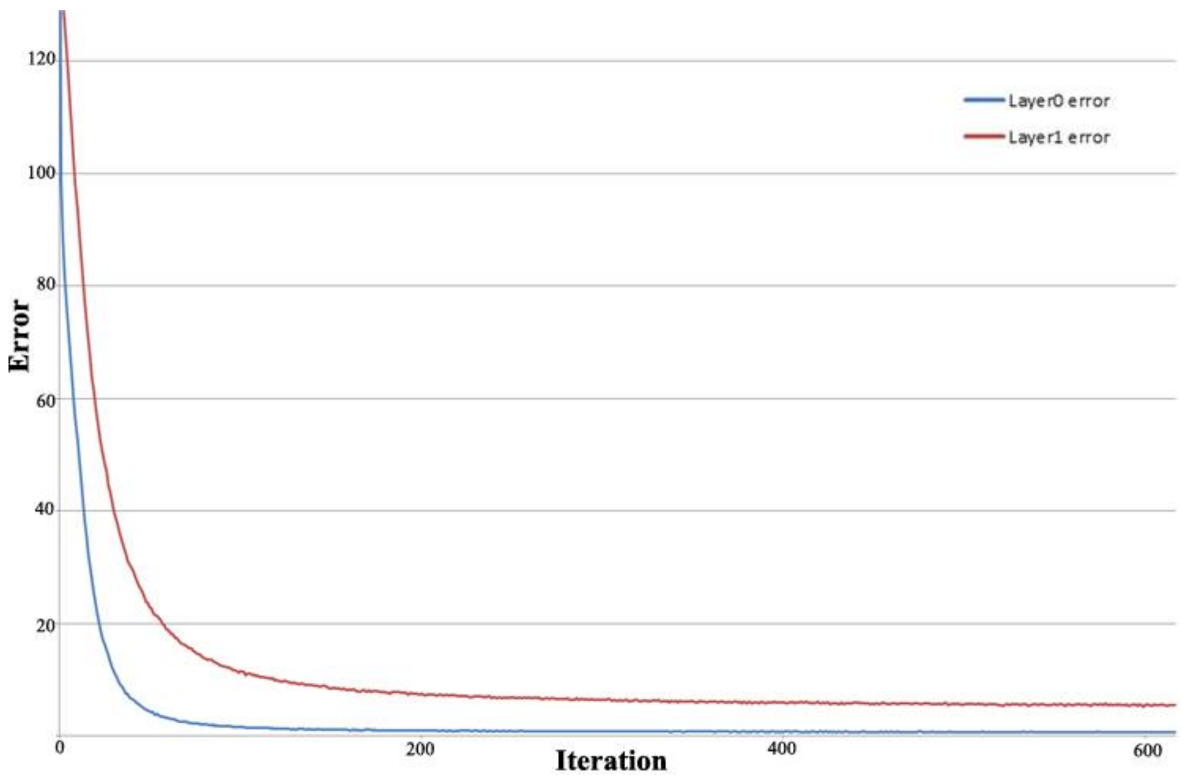


Figure 4-4 Segment network training.

4.3. Clustering

Once the training for the networks is done, a set of 300 songs have been computed through the networks and the activation vectors are written to a file for MATLAB function to process. MATLAB have provided more reliable framework for clustering analysis and have given better performance in terms of calculation speed along with more features, thus I have decided to use MATLAB for this step after comparing a self-implemented k-means algorithm on C#.

I have decided the segments are the basis of these three relations, a pitch or loudness loosely depends on the duration of a note. Thus, using the dimensionality reduction toolbox in Matlab, I've implemented a K-means clustering algorithm that prioritizes segments and then clusters the pitch and loudness hierarchically inside those clusters.

The algorithm finds 10 clusters in segment network vectors, then inside those clusters tries to find 5 clusters for each pitch and loudness vectors. Resulting in total of 10 segment clusters and 50 (if exists) pitch and loudness clusters.

I have written the cluster means and matching relations to a file. This way I can use a relational fitness function based on segment, pitch and loudness or I can ignore it and use them individually.

4.4. Generation

I have followed the basic framework during the development of my genetic algorithms. Class hierarchy and C# implementation is highly influenced by AForge library [15] for consistency.

4.4.1. Genetic Algorithm

A Chromosome.cs class is implemented to hold the value of note strings as an array of unsigned short (16 bit), fitness value and carry out basic operators of crossover and mutation. Chromosomes can be initialized randomly or with a seed value, which puts a bias towards a certain area in search space if required.

Desired amount of these chromosomes are generated and kept in Population.cs class where the genetic algorithm steps are applied in order. A flowchart of these steps that are implemented by the framework can be seen in Figure 4-5.

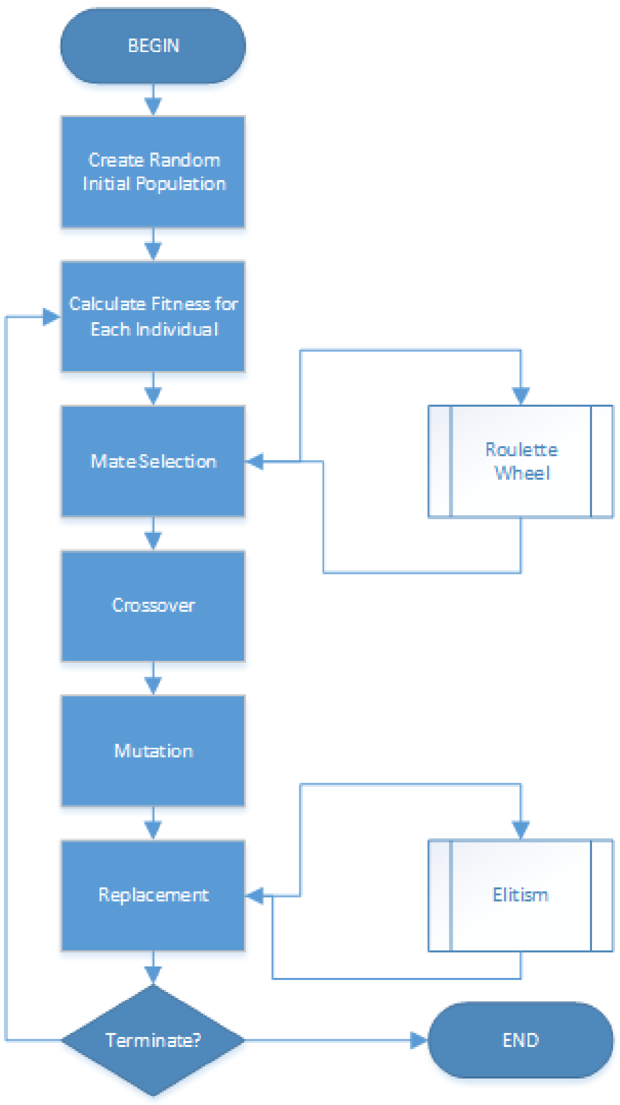


Figure 4-5 Genetic algorithm implementation.

One point crossover operator and shotgun mutation method is used where a new individual is introduced with a mutation rate checked over chromosome itself.

Multiple mate selections methods are tested and available in project including rank selection, tournament selection etc. but roulette wheel selection seemed to be most promising in terms of convergence during tests, thus used as the main mate selection method throughout the project.

Interface pattern have been used on selection and fitness function classes for consistency. Abstract classes have been used throughout for ease of implementation.

4.4.2. Fitness Function

Multiple variations of hamming distance based maximizing fitness functions have been implemented for the networks. These fitness functions can be used in conjunction or individually by simply summing up the fitness values since the outputs are in same range. These fitness functions are combinations of 3 base topics;

- Network relation: The fitness evaluation can be done using the said hierarchical relation between segment, pitch and loudness clusters in k-means algorithm. If the relation parameter is set, then the pitch and loudness evaluations are only calculated in their respective segment clusters.
- Population type: A possible scenario is a dominance of a network cluster and crowding, especially when network relation is used since the chromosome would converge towards pitch and loudness clusters inside a segment cluster after a certain time and it may start to favor that segment cluster over others. To prevent this I have implemented a parallel population algorithm, where 10 different populations would initialize and run at the same time in 10 different segment clusters separately. Algorithm would give the best fitness value as a result.
- Cluster threshold: Due to the limited representation of my model mean values may not be represented properly; hence songs generated in my algorithm converge, but may not get good fitness values. Thus I've reconstructed the cluster means in the solution domain, applied a threshold function to fit the reconstructed input into project's model and reiterated these proper mean representations back into the network for a solution that a chromosome can better converge. This parameter can be set to use these proper mean representations of cluster means.

I have experimented with these settings and have not seen a significant difference between cluster threshold parameter. While network relation only affected a little, parallel run of the algorithm have improved the performance significantly.

A single population, direct network output pseudocode for the fitness function that respects the named hierarchical relation is given in Table 4-4.

Table 4-4 Pseudocode for single population, relational fitness function without threshold.

Algorithm 3 Pseudocode for single population relational fitness function

```
function fitness(chromosome)
    inputs: chromosome, array of ushort values
    chromosomeSegment ← encodeSegments(chromosome);
    chromosomePitch ← encodePitch(chromosome);
    chromosomeLoudness ← encodeLoudness(chromosome);
    fitness = -∞, k = 0;
    for i ← 0 to segmentCount do
        intermediate fitness = 0;
        for j ← 0 to chromosomeSegment.Length do
            intermediateFitness += Abs(chromosomeSegment[j] – segments[i][j]);
        endfor
        if(intermediateFitness > fitness)
            fitness = intermediateFitness;
            k = i;
        endif
    endfor
    fitnessPitch = -∞;
    for i ← 0 to pitch[k].Count do
        intermediateFitness = 0;
        for j ← 0 to chromosomePitch.Length do
            intermediateFitness += Abs(chromosomePitch[j] – pitch[k][i][j]);
        endfor
        if(intermediateFitness > fitnessPitch)
            fitnessPitch = intermediateFitness;
        endif
    endfor
    fitnessLoudness = -∞;
    for i ← 0 to loudness[k].Count do
        intermediateFitness = 0;
        for j ← 0 to chromosomeLoudness.Length do
            intermediateFitness += Abs(chromosomeLoudness[j] – loudness[k][i][j]);
        endfor
        if(intermediateFitness > fitnessLoudness)
            fitnessLoudness = intermediateFitness;
        endif
    endfor
    return fitness + fitnessPitch + fitnessLoudness;
endFunction
```

Finally, I have decided to use a population size between 100 – 200 and maximum iteration count between 750 – 1250 using my intuition, based on an individual chromosome length, which is equal to the size of the solution space. Chromosome length can be calculated using the Equation 4-2 below.

$$sSize = cLength = \#notes \times 2^{16} \qquad (Equation\ 4-2)$$

As decided earlier on analysis state, I have generated multiple outputs using large mutation rates ranging between 0.1 to 0.3 and 10% elitism.

Progression of fitness value during first 400 iterations with following parameters (threshold:false, relational:false, population:single) and population size of 150 can be seen in Figure 4-6 below.

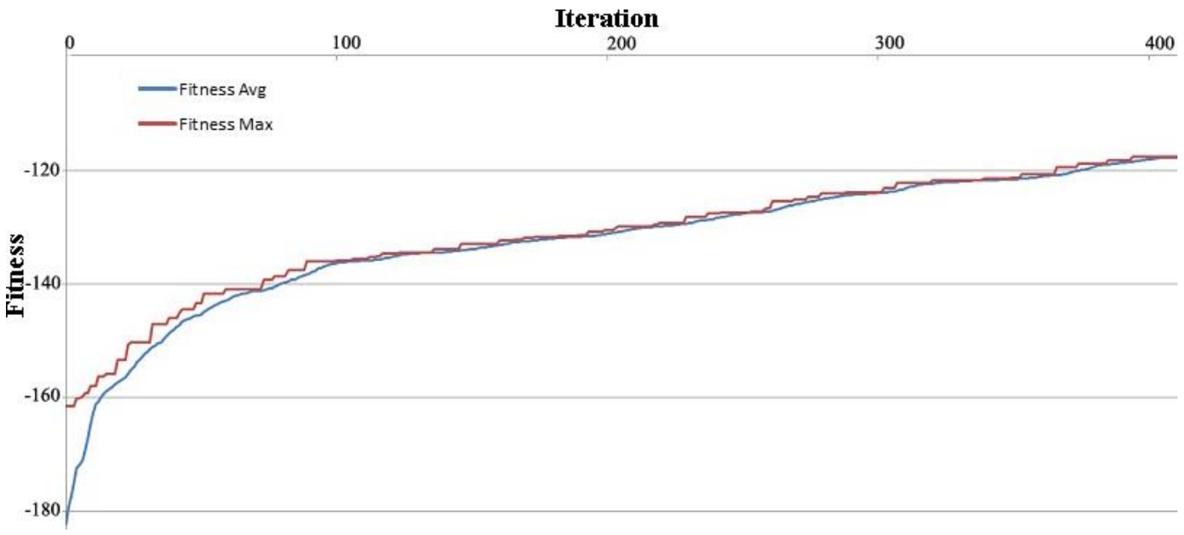


Figure 4-6 Fitness by time of single population without threshold and relational clustering.

Progression of fitness value during first 400 iterations with following parameters (threshold:true, relational:true, population:parallel) and population size of 40 in each cluster for a total of 400 can be seen in Figure 4-7. Total population size has been increased in parallel run because; populations are now separated to 10 individual groups and require genetic diversity for better convergence, which cannot be achieved using an individual population size of 15.

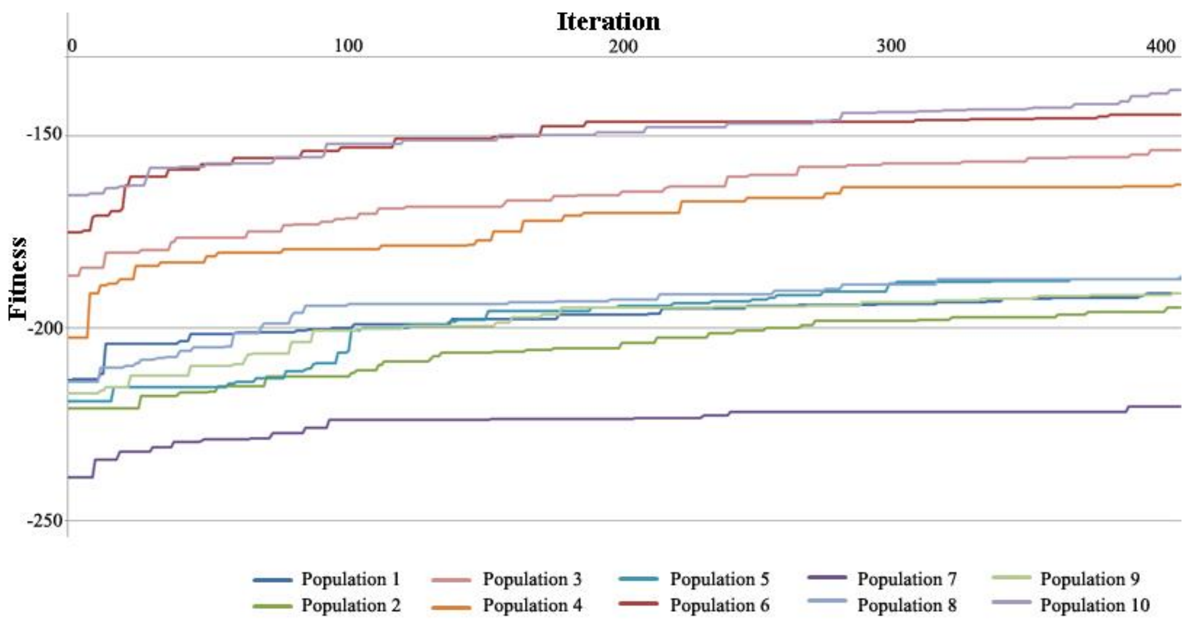


Figure 4-7 Fitness by time of parallel population using threshold and relational clustering.

5. TEST AND RESULTS

Evaluating computer generated music has difficulties in measuring the effectiveness and success rate, there are formula’s or rules in theory but they only cover basics and do not fare well with the final opinion where the subjects are humans.

Thus evaluation must be made where the decision maker is human audience, but that too has problems, music domain is a subjective matter where different tastes and level of experience in domain may affect the outcome. Hence tests must be made in an objective manner with multiple subjects and resulting data must be read according to the audience’s level of experience.

5.1. Testing Environment

Test have been constructed and applied in a similar manner to the earlier tests practiced on computer generated music. A version of RENCON test for evaluation of music generation [16] that is used at the 2011 Sound and Music Computing conference has been taken as basis with some additions from Evaluating Musical Metacreation in a live Performance [17].

Total number of 20 students from Yeditepe University Computer Engineering department has volunteered to participate in tests. Volunteers have been asked to fill an anonymous form with the help of a supervisor for data integrity. Volunteers have been asked to rate their musical understanding as a general listener and ear training as well as music theory knowledge in a Likert scale for an understanding of their level of experience in music domain.

Then, the volunteers are presented with three different songs; an output of the algorithm, a neural network fantasy of the cluster means and a random note strings in no peculiar order. Where algorithm outputs are genetic algorithm results, using various combinations of discussed fitness function parameters. And fantasies are the reconstructions of randomly selected cluster means using the symmetric weight distribution of restricted Boltzmann machines, a method similar to what has been used in alternate Gibbs sampling, where network is activated from output layer to input layer. Songs were midi outputs of chromosomes in acoustic piano for better understanding. After listening to each piece, volunteers have been asked to rate the songs on following questions using a Likert scale ranging 1 to 5;

- Rate the composer of the song you have listened on a scale of very weak (first time seeing a piano) to very strong (expert composer).
- Rate how engaging you have found the song from very weak to very strong.

After listening to all three songs, volunteers are also asked to guess which song belonged to the random note strings.

This evaluation process has been iterated 4 times over where total of 12 songs played and volunteers are asked to comment on what they thought about the songs, especially what genre they think the songs belong to.

A small twist has also been added to the test to set the scaling and better evaluation of the generated songs; final generated song was a low fitness song, having similar structure to one of the songs played earlier with the difference of not being in key. Midi output of that said song was constructed with additional bassline with same notes and string quartet with same notes to create a very basic polyphony. Votes on this song were significantly above average and some volunteers have also commented on this song being the best of all.

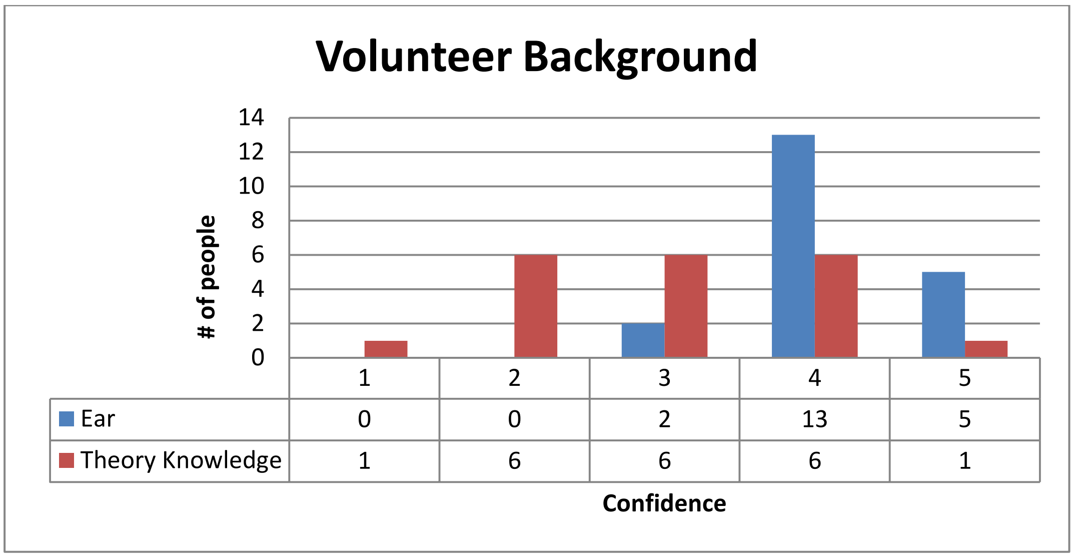
5.2. Test Results

Following two questions asked to have a better understanding of the participants;

- Rate yourself from 1 to 5 in terms of your general knowledge and musical ear.
- Rate yourself from 1 to 5 in terms of your knowledge about Music theory knowledge (note, scale, chord progression etc.)

Out of 20 volunteers an evenly distribution of participants with music theory education background was observed while most of the participants evaluated themselves as strong listeners which can be seen in Figure 5-1.

The difference in ratings over the songs, between the general population and the participants with better ear or prior music theory knowledge was mostly insignificant.



	1	2	3	4	5
🟦 Ear	0	0	2	13	5
🟥 Theory Knowledge	1	6	6	6	1

Figure 5-1 Volunteer's background on music.

Volunteers are then represented with songs from the first experiment in random order and asked to answer the following questions for each song;

- Rate the composer of the song you have listened on a scale from very weak (first time seeing a piano) to very strong (expert composer).
- Rate how engaging you found the song from very weak to very strong.

Figure 5-2 shows the volunteers' votes; where labels below a bar cluster represent the question for a given algorithm, for example Genetic1 label cluster represents volunteers' ratings to genetic algorithm output for question 1 (rate the composer) in a scale of 1 to 5, Fantasy1 label represents volunteers' ratings to fantasy reconstruction for question 1 etc. Blue bars on a cluster represent average ratings, whereas red bars represent the weighted average ratings with respect to volunteers' musical ear and green bars represent the weighted average ratings with respect to volunteers' music theory knowledge.

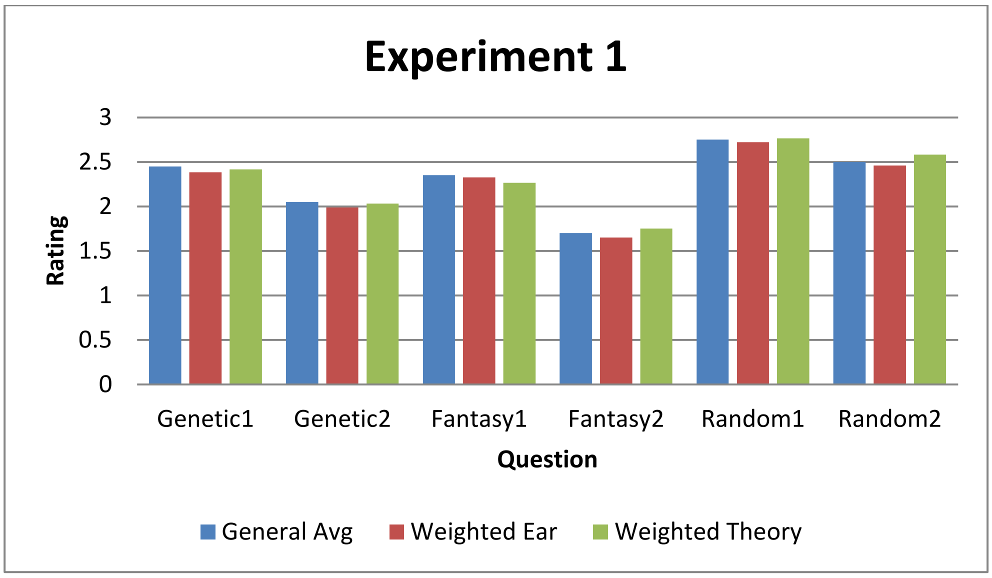


Figure 5-2 Experiment 1 results.

Same process has been iterated three more times, for a total of 12 songs. Following figures with same representation shows the results of said experiments in order.

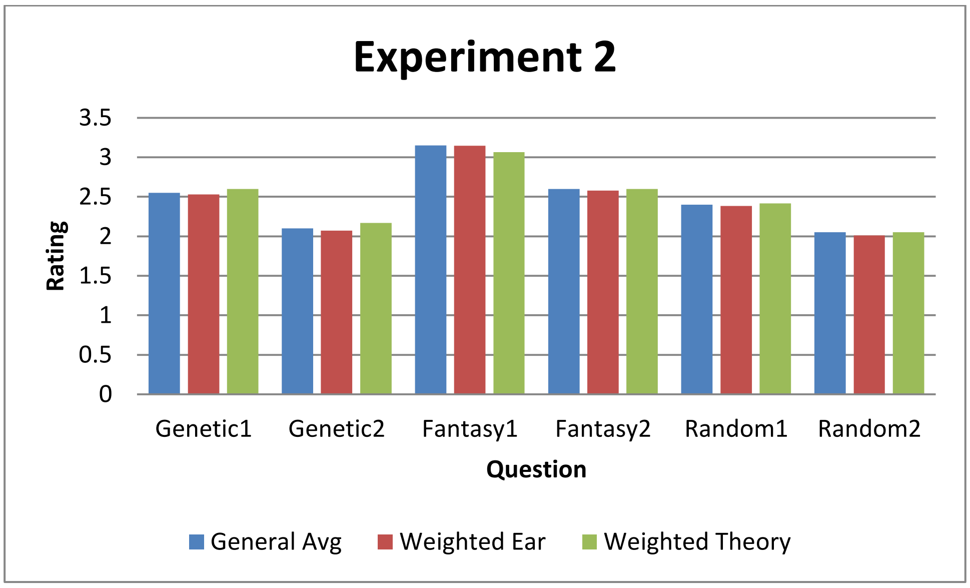


Figure 5-3 Experiment 2 results.

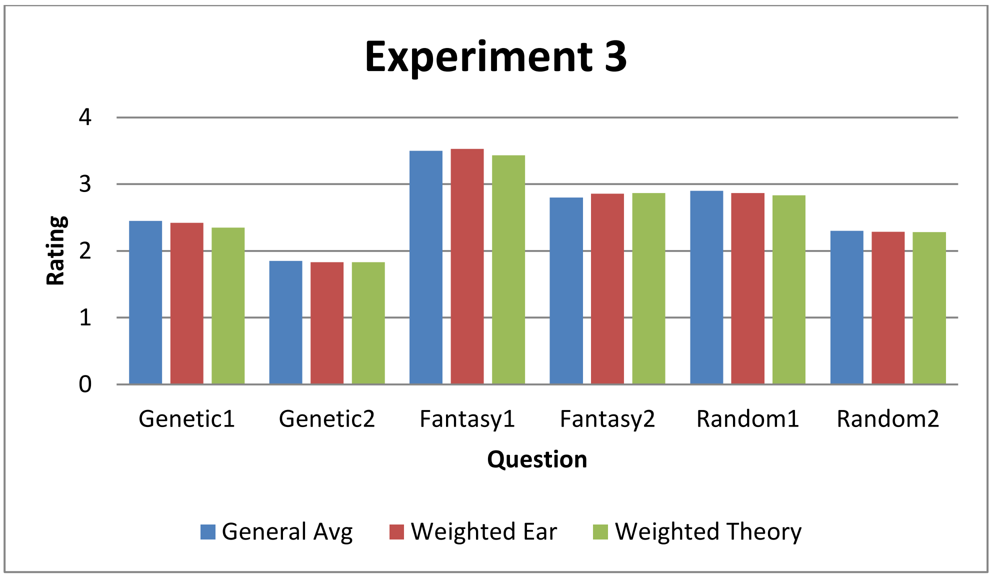


Figure 5-4 Experiment 3 results.

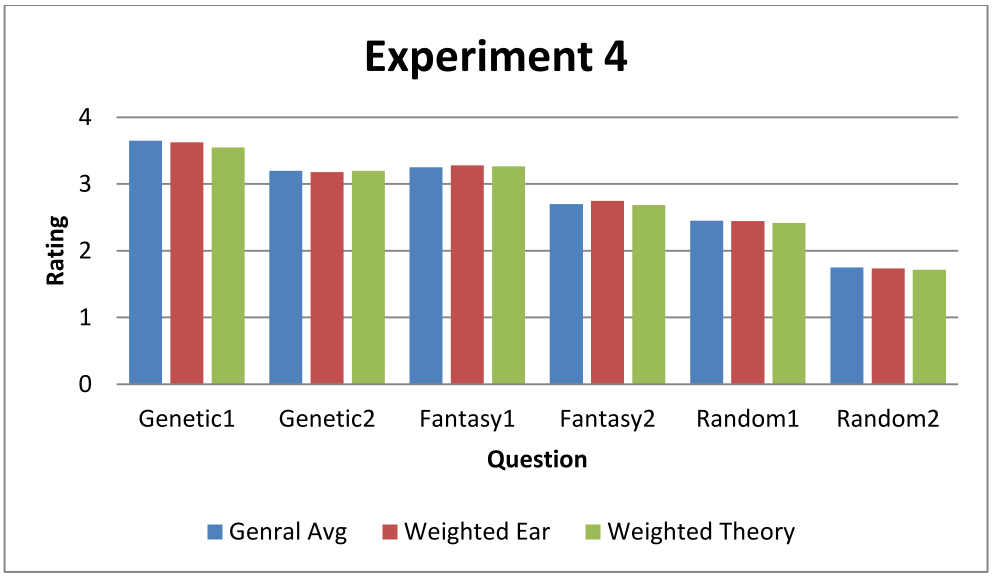


Figure 5-5 Experiment 4 results.

Genetic algorithm solution in the experiment 4 included bass and violin with the same note and chord progression in the midi output, which have created a simple polyphony. This has affected the results significantly in positive direction despite the solution not being in key. When the graph analyzed with respect to the theory knowledge of the participants, it can be seen that this was more effective on general population rather than trained participants.

And the average results for all experiments can be seen in Figure 5-6 below.

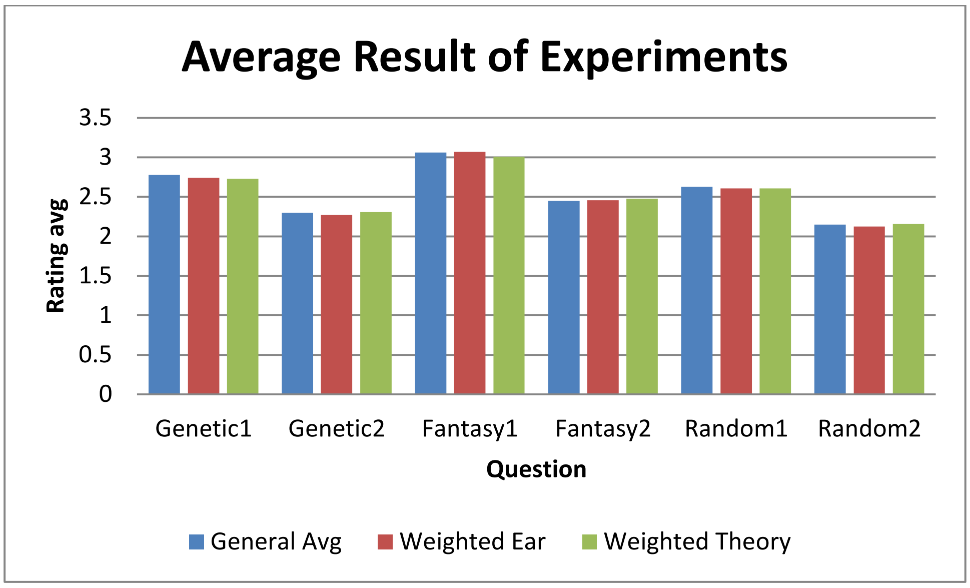


Figure 5-6 Overall experiment results.

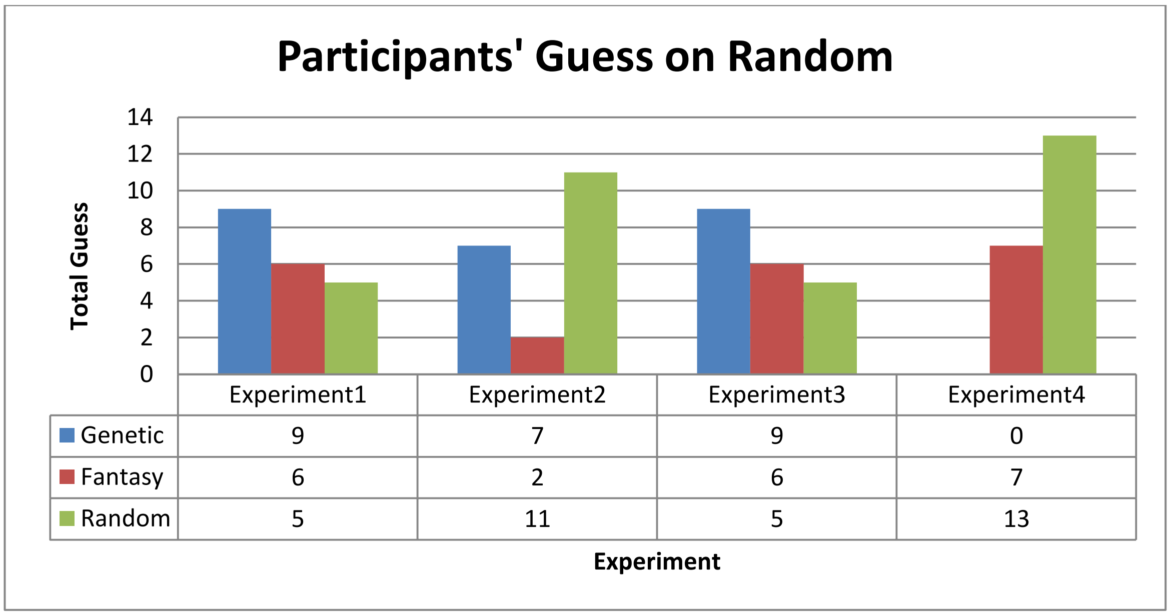
Looking at the overall results, the output of the project did not perform better than the random note strings according to participants, while neural network reconstructions were more professional and slightly more interesting.

Furthermore the results tested were generated using four of the different fitness function combinations. Participants did not seem to have a significant preference.

Analyzing the comments and the reactions, these negative results may be due to the bad interpretation of segment values from the dataset. Genetic algorithm results were mostly slower songs combined with notes that prefer long intervals, when listened with a single instrument it led listeners to think it was not interesting, which have been fixed using simple polyphony in experiment 4. While neural network results were dull in terms of pitch and loudness, they had a tendency to play in faster intervals of segments.

Although segment training was not as interesting, information gain from pitch and loudness networks was significantly better as stated from the participants' comments.

Volunteers were also asked guess which song composed of random notes at the end of each experiment. The volunteers' guesses for all experiments can be seen in Figure 5-7, where bar clusters represent an experiment in which; genetic algorithm votes are represented with blue, fantasy votes (cluster mean reconstructions) are represented with red and random note string votes are represented with green.



	Experiment1	Experiment2	Experiment3	Experiment4
🟦 Genetic	9	7	9	0
🟥 Fantasy	6	2	6	7
🟩 Random	5	11	5	13

Figure 5-7 Participants guess' on random songs.

On the most experiments, volunteers were able to distinguish the fantasies easily, while genetic algorithm output was only slightly more distinguishable than random output.

Participants have also commented that the most of the songs had some kind of structure. More specifically some number of participants has commented that most of the songs were following the correct theoretical chord progression, obeying the scale and had a key. This concludes that training was successful to extract the intended features from the dataset and this information is carried during clustering to genetic algorithms for successful output.

Participants were also encouraged to comment on the genre of the music they listened to. Most of participants commented towards a hybrid genre of jazz, blues and classical music. Instrument used in the experiments, which was acoustic grand piano for easy listening was putting a bias towards classical music as later discussed with participants.

6. CONCLUSION

During the project a feature set of duration, loudness and pitch from songs were successfully extracted and used in different combination to generate solutions.

Furthermore, multiple populations and custom relation settings for clusters have been experimented to develop an optimal fitness function for music domain. Although, according to test results these different combinations did not differ in a significant way, results have proven that the clustered information extracted from the dataset using stacked RBM’s was successful to create a basic framework for the fitness function.

While duration was not able to generate interest in volunteers, pitch and loudness information gained from the networks was most beneficial to the system.

Due to the complex nature and large solution space of music domain, projects correlations and information gained from the dataset was too primitive, hence project was not successful at producing human quality compositions.

As a result project’s output was either too boring to listen or too creative, it feels random. This can be either fixed using more information, which results in somewhat over-fitted songs or using a mentor program in fitness function that uses music theory, which reduces the autonomy.

7. FUTURE WORK

The project's framework; learning fitness function, using neural networks and clustering can be applied to other domains where a vague or unknown fitness function problem exists. The project may even achieve better results in computational painting, where feature extraction methods have advanced greatly over the last decade.

7.1. Computational Painting

Using the same, three main stages used in this project; any art form can be learned and used to generate from average features. Especially after the implementation and test results, computer generated pictures seems to be a more fitting domain due to better representation in pixel data and feature extraction using convolutional neural networks.

Convolutional neural networks can be used to encode a painting into a set of filter activations and feature representation of that painting can be a set of activation matrixes. These features can be correlated by filter to pixel basis, reduced to smaller spaces using RBM's. The outputs are then clustered again to get the mean average of a painting and used in a hamming distance based fitness function in genetic algorithms. A similar flowchart used in the project is designed below in Figure 7-1 for a future project.

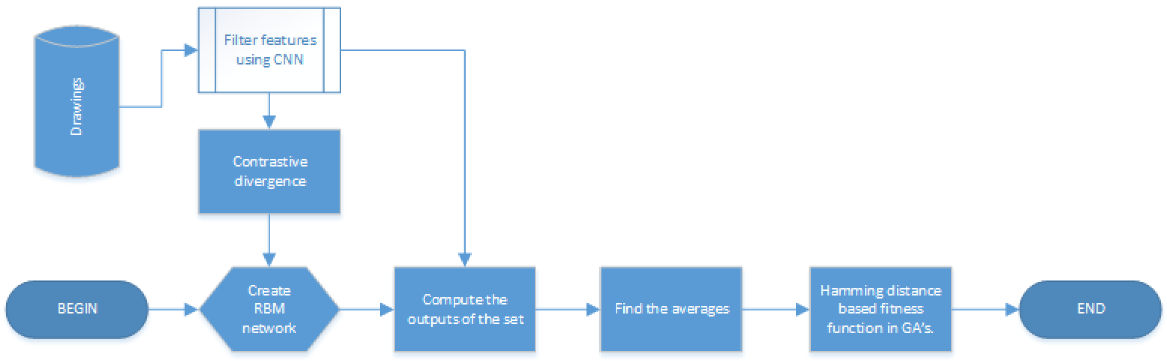

Drawings
Filter features using CNN
Contrastive divergence
BEGIN
Create RBM network
Compute the outputs of the set
Find the averages
Hamming distance based fitness function in GA's.
END

Figure 7-1 Flowchart for computational painting algorithm.

7.2. Benefits in Musicological Studies

Since the project can extract the prominent features of a given dataset and find the basis structural average of given songs. This information may be valuable in musicology area. Especially in ethnomusicological practices, where the music of a particular repertoire is studied in a cultural context.

Project can be activated by using a repertoire of specific area to be studied and cluster mean average would produce valuable information about the musical structure for that area. This structure may be used to study patterns and features or may be used for categorization of songs in that context. Similarly, using genetic algorithms upon this contextual basis, new songs can be created in cultures where composers cease to exist.

BIBLIOGRAPHY

[1] Johanson, Brad, and Riccardo Poli. GP-music: An interactive genetic programming system for music generation with automated fitness raters. University of Birmingham, Cognitive Science Research Centre, 1998.

[2] Tollervey, Nicolas. Foox - Music Theory, Genetic Algorithms ("Ntoll/foox." GitHub). 24 Dec. 2015. <https://github.com/ntoll/foox>.

[3] Dostál, Martin. "Genetic Algorithms As a Model of Musical Creativity--on Generating of a Human-Like Rhythmic Accompaniment." Computing and Informatics 24.3 (2012): 321-340.

[4] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

[5] Hinton, Geoffrey E. "Training products of experts by minimizing contrastive divergence." Neural computation 14.8 (2002): 1771-1800.

[6] Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1 (2009): 1-127.

[7] Biles, John. "GenJam: A genetic algorithm for generating jazz solos." Proceedings of the International Computer Music Conference. INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1994.

[8] Bickerman, Greg, et al. "Learning to create jazz melodies using deep belief nets." First International Conference on Computational Creativity. 2010

[9] Boulanger-Lewandowski, Nicolas, Yoshua Bengio, and Pascal Vincent. "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription." arXiv preprint arXiv:1206.6392 (2012).

[10] Sutskever, Ilya, Geoffrey E. Hinton, and Graham W. Taylor. "The recurrent temporal restricted boltzmann machine." Advances in Neural Information Processing Systems. 2009.

[11] Bertin-Mahieux, Thierry, et al. "The million song dataset." ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida. University of Miami, 2011

[12] Wang, Kaijun. "(simple) Tool for Estimating the Number of Clusters." 25 Dec. 2015. <http://www.mathworks.com/matlabcentral/fileexchange/13916--simple--tool-for-estimating-the-number-of-clusters>.

[13] van der Maaten, Laurens JP, Eric O. Postma, and H. Jaap van den Herik. "Dimensionality reduction: A comparative review." Journal of Machine Learning Research 10.1-41 (2009): 66-71.

[14] Souza, Cesar. "Accord.NET Machine Learning Framework." Accord.NET Machine Learning Framework. Web. 25 Dec. 2015. <http://accord-framework.net/>.

[15] "AForge.NET :: Framework." AForge.NET :: Framework. Web. 25 Dec. 2015. <http://www.aforgenet.com/framework/>.

[16] Katayose, Haruhiro, et al. "On evaluating systems for generating expressive music performance: the rencon experience." Journal of New Music Research 41.4 (2012): 299-310.

[17] Eigenfeldt, Arne, Adam Burnett, and Philippe Pasquier. "Evaluating musical metacreation in a live performance context." Proceedings of the Third International Conference on Computational Creativity. 2012.