

**GenrELM: An ELM Approach to Automatic Music Genre
Classification using Bass Lines**

Presented to the Faculty of the
Natural Sciences and Mathematics Division
University of the Philippines Visayas
Tacloban College

In Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Science in Computer Science

Presented By:
J. Gabrielle M. Macaraig
2007-34203

April 2013

Approval Sheet

This project entitled “GenrELM: An ELM Approach to Automatic Music Genre Classification using Bass Lines” by J. Gabrielle M. Macaraig was presented to the Faculty of the Natural Sciences and Mathematics Division of the University of the Philippines Visayas Tacloban College.

This project is hereby accepted and approved in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

John Paul T. Yusiong

Adviser

Date

Acknowledgement

My deepest gratitude to the following people who have shared their time and knowledge as I make this research:

To my adviser, Prof. John Paul T. Yusiong, for sharing not only his knowledge and expertise, but his time and life as well, and though we disappoint him many times, yet he continued to believe in us and give us numerous chances and considerations that we may have a chance of graduating in April.

To Ma'am Anni, Sir Victor, Sir Ryan, Sir Dave, Ma'am Blez, Ma'am Ritch, and to the rest of my teachers, for imparting knowledge that have been very useful not only to this research, but in the real world as well.

To my family, for their love, care and understanding, and for providing me the necessary resources I need in making this research possible.

To Kuya Jake and Reeno for their encouragements when I felt like giving up, and to James, who had not only encouraged me, but also shared his time and knowledge as we work on our SP.

To all my classmates, for keeping me company and for the friendship we've shared all throughout my college days. Thank you guys for the laughter and happy moments that had cheered me up especially when I was feeling down because of this SP.

To Dr. Simsekli and Dr. McKay, who have given me useful tips for my research and most especially, for responding to my emails despite their busy schedule.

To Kuya Glyndon and Nitoy, for giving useful tips, for sharing their experiences during the time they were working on their own SP's, and for encouraging me that I, too, can finish my SP.

Above all and most especially, I would like to give my praise and thanks to the Lord God Almighty, the source of my wisdom, my strength and my all; the Mighty One who has made me conquer all the challenges I have encountered especially in this research, and who has been my help most especially in times of need. His grace gave me strength to face each new day with hope, even though brighter days had seemed far. In my weaknesses, He became my strength and my refuge, and in times when I lack, He has provided for me sufficiently. In times when I was sinking in hopelessness, His mighty hand took me out of my sea of trouble despite my lack of faith. Glory be to God, whose promises are true and whose love never fails, for making this research possible, and for making me reach this far.

Abstract

The Extreme Learning Machine, or ELM, is a machine learning algorithm for classification problems. Similar to artificial neural networks, ELM learns by example, except that it learns without the need for iterative tuning. It also addresses some issues common in ANNs such as slow learning speed, over-training, and local optima entrapment. In this study, the ELM is used for automatic music genre classification using information extracted from bass lines, particularly the melodic information. A bass line is a monophonic instrumental melody usually made distinct from other sounds in a musical piece by its characteristic of being low-pitched. Furthermore, it is said that bass lines, in most musical styles, are able to link together melodic and rhythmic elements, thus, it is a rich source of information for music genre classification. Experimental setups in this study include testing MIDI files using the ELM with and without a regularization parameter. Using a genre taxonomy having 3 root genres and 9 root genres, results show that the ELM classifier achieved a high classification accuracy rate. Moreover, the ELM with a regularization parameter performed better than the ELM without a regularization parameter.

Keywords: Extreme Learning Machine, Music Genre Classification, Bass Line, MIDI, Melodic Information

Contents

List of Figures	3
List of Tables	5
1 Introduction	7
1.1 Music 101: Knowing the Basics	8
1.1.1 Elements of Music	8
1.1.2 Musical Instruments	9
1.1.3 Genre	11
1.2 Bass Line	12
1.3 Music Data	12
1.3.1 High-level Music Features	13
1.3.1.1 Melodic Intervals	13
1.4 MIDI Format	14
1.5 Machine Learning	15
1.6 Neural Networks	15
1.6.1 Neurons in the Real World	15
1.6.2 The Artificial Neuron and ANN	17
1.6.3 Network Structures	18
1.6.4 Learning and Classification	19
1.7 Extreme Learning Machine	19
1.7.1 Three-step Learning Algorithm	20
1.7.2 ELM with Regularization Parameter	22
1.7.3 Learning without Iterative Tuning	22
2 Literature Review	23
3 Statement of the Problem	28
4 Objectives	29
5 Proposed Approach	30
5.1 Overview of the Approach	30

5.2	Input	30
5.3	Input Representation and Feature Extraction	31
5.4	Genre Taxonomy	32
5.5	Data Set	32
5.6	Training with ELM	32
5.6.1	Training Set	32
5.6.2	Network Structure	33
5.6.3	Input Nodes	33
5.6.4	Nodes in the Hidden Layer	33
5.6.5	Output Nodes and Target Output Mapping	34
5.6.6	Extreme Learning Machine Training Flow	35
5.7	Testing the Network	36
5.8	Output	39
6	Experiments and Results	40
6.1	System Specifications	40
6.2	Experimental Settings	40
6.3	Experimental Results and Discussions	42
6.3.1	Variation in Training and Test Sets Used	42
6.3.2	Variation in Number of Hidden Nodes	48
6.3.3	Introduction of a Regularization Coefficient	55
6.3.4	Variation in Network Structure	60
6.3.5	Comparison with Existing Study	64
7	Conclusion and Future Works	66
	References	68
	Appendix: User's Manual	71

List of Figures

1.1	Instruments in an Orchestra	10
1.2	A Typical Rock Band Setup	11
1.3	Illustration of Semitones in a Piano	14
1.4	The Biological Neuron	16
1.5	The Artificial Neuron	17
1.6	Multilayer Perceptron	18
1.7	Structure of a Single-Hidden Layer Feed-forward Neural Network	20
5.1	Program Flow	30
5.2	Feature Extraction and Input Representation	31
5.3	The ELM Network Structure for GenrELM	33
5.4	ELM Training Flow	35
5.5	Hidden Node Parameter Generation	35
5.6	Root Genre Classification on <i>4on6[bass].mid</i>	36
5.7	Leaf Genre Classification on <i>4on6[bass].mid</i>	37
6.1	Accuracy Rate on Variation in Training and Test Sets Used – Test Set (Root)	44
6.2	Accuracy Rate on Variation in Training and Test Sets Used – Test Set (Jazz)	45
6.3	Accuracy Rate on Variation in Training and Test Sets Used – Test Set (R'nB)	46
6.4	Accuracy Rate on Variation in Training and Test Sets Used – Test Set (Rock)	47
6.5	Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Root)	50
6.6	Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Jazz)	51
6.7	Accuracy Rate on Varying Number of Hidden Nodes – Test Set (R'nB)	52

6.8	Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Rock)	53
6.9	Accuracy Rate on Varying Regularization Coefficients – Test Set (Root)	56
6.10	Accuracy Rate on Varying Regularization Coefficients – Test Set (Jazz)	57
6.11	Accuracy Rate on Varying Regularization Coefficients – Test Set (R'nB)	58
6.12	Accuracy Rate on Varying Regularization Coefficients – Test Set (Rock)	59
6.13	Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Flat Classification)	62
6.14	Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Flat Classification)	63

List of Tables

5.1	Genre Taxonomy Adopted in GenrELM	32
5.2	Target Output Encodings for Root Genres	34
5.3	Target Output Encodings for Jazz Leaf Genres	34
5.4	Target Output Encodings for Rhythm ‘n Blues Leaf Genres	34
5.5	Target Output Encodings for Rock Leaf Genres	35
5.6	Maximum Output Node-to-Genre Mapping for Root Genres	38
5.7	Maximum Output Node-to-Genre Mapping for Jazz Leaf Genres	38
5.8	Maximum Output Node-to-Genre Mapping for Rhythm ‘n Blues Leaf Genres	38
5.9	Maximum Output Node-to-Genre Mapping for Rock Leaf Genres	38
6.1	Dataset Partitioning for GenrELM	41
6.2	Variation in Training and Test Sets Used (Training Set) – Root Results	43
6.3	Variation in Training and Test Sets Used (Test Set) – Root Results	44
6.4	Variation in Training and Test Sets Used (Training Set) – Jazz Results	45
6.5	Variation in Training and Test Sets Used (Test Set) – Jazz Results	45
6.6	Variation in Training and Test Sets Used (Training Set) – R’nB Results	46
6.7	Variation in Training and Test Sets Used (Test Set) – R’nB Results	46
6.8	Variation in Training and Test Sets Used (Training Set) – Rock Results	47
6.9	Variation in Training and Test Sets Used (Test Set) – Rock Results	47
6.10	Variation in Number of Hidden Nodes (Training Set) – Root Results	49
6.11	Variation in Number of Hidden Nodes (Test Set) – Root Results	50
6.12	Variation in Number of Hidden Nodes (Training Set) – Jazz Results	51
6.13	Variation in Number of Hidden Nodes (Test Set) – Jazz Results	51
6.14	Variation in Number of Hidden Nodes (Training Set) – R’nB Results	52
6.15	Variation in Number of Hidden Nodes (Test Set) – R’nB Results	52
6.16	Variation in Number of Hidden Nodes (Training Set) – Rock Results	53
6.17	Variation in Number of Hidden Nodes (Test Set) – Rock Results	53

6.18	Variation in Regularization Coefficient (Training Set) – Root Results	55
6.19	Variation in Regularization Coefficient (Test Set) – Root Results	55
6.20	Variation in Regularization Coefficient (Training Set) – Jazz Results	56
6.21	Variation in Regularization Coefficient (Test Set) – Jazz Results	57
6.22	Variation in Regularization Coefficient (Training Set) – R'nB Results	58
6.23	Variation in Regularization Coefficient (Test Set) – R'nB Results	58
6.24	Variation in Regularization Coefficient (Training Set) – Rock Results	59
6.25	Variation in Regularization Coefficient (Test Set) – Rock Results	59
6.26	Basic ELM vs. ELM w/ Reg. Coeff – Average Accuracy	60
6.27	Variation in Number of Hidden Nodes (Training Set) – Flat Classification Results	61
6.28	Variation in Number of Hidden Nodes (Test Set) – Flat Classification Results	62
6.29	Variation in Regularization Coefficients (Training Set)– Flat Classification Results	63
6.30	Variation in Regularization Coefficients (Test Set) – Flat Classification Results	63
6.31	KNN vs. ELM Genre Classification Results	64

Chapter 1

Introduction

From observation alone, one can deduce that man and music is inseparable, as music is said to be a form of expression of one's self, and that man is greatly inclined towards music. People's inclination towards music is so evident that we see people having earphones plugged into their ears, listening to their favorite songs stored in their mobile phones or MP3 players, or even having music being played in the background as we enter stores, coffee shops and restaurants or even as we ride in vehicles. We even know people who have their music players playing in the bathroom while they take a bath, or have them playing in the background to help them fall asleep. Music is undoubtedly a normal part of our everyday lives.

Different people have different preferences or tastes on genre or the kind of music they want to hear. Some people want to listen to the relaxing sound of smooth jazz, while others want to bang their heads along the rhythm of rock music. Some people, on the other hand, enjoy listening to classical music and even listen to them as they study for their exams, while others have a bias towards music under the R'n B genre.

Due to these existent genre preferences among different people of all ages, and due to the technological advent we are in today, there have been many attempts to produce an automatic system for music genre classification that will aid both the market and consumers in their music tastes in many ways, such as in music database organization.

Different approaches were employed in developing an automated music genre classification system. Algorithms such as Support Vector Machines, Hidden Markov Models, and K-Nearest Neighbour classifiers were among those that were used in different researches and experiments. Different feature sets were also utilized such as

Mel-frequency cepstral coefficients. In this paper, bass lines and the ELM algorithm will be used in solving the music genre classification problem.

1.1 Music 101: Knowing the Basics

According to Merriam-Webster, music is “the science or art of ordering tones or sounds in succession, in combination, and in temporal relationships to produce a composition having unity and continuity [20].” Based on the given definition, music can be said as the end product of a mixture of elements such as tone and timing, perfectly blended altogether thus having unity and continuity. The study of music as a field of knowledge is called *musicology*.

1.1.1 Elements of Music

There are four basic elements of music: rhythm, melody, harmony, and texture [22]. Other references include dynamics and tone color [3] [4].

Rhythm refers to the music’s element of “time”. It can also be technically described as the flowing movement of sound with repeating heavy and/or light beats at regular intervals. It is usually this element that makes one move along with music, such as tapping of one’s feet, the clapping of hands, the banging of heads, and even dancing. Never did we hear the phrases “dance according to the melody”, or “move according to the harmony or texture of the music”. Rhythm can be quantifiable in terms of duration tempo, or meter. [3] [22]

Melody, on the other hand, refers to the linear or horizontal presentation of the highness or lowness of a musical sound referred to as the *pitch* [3]. In a more understandable fashion, the melody is the tune or the singing part or the hummable part in a musical composition. It comprises of many single tones having different levels of pitch. *Scales* and *keys* are two concepts in music theory that are associated with this element of music.

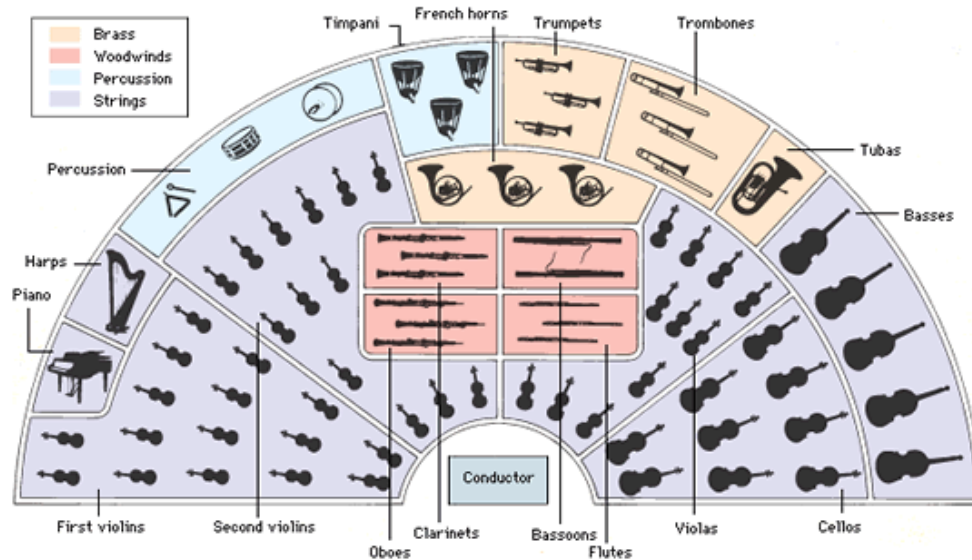
Harmony, in contrast to melody, is the vertical presentation of pitch. It is usually imagined as the art of using chords – its structure, relationship and progression. A *chord* is formed by three or more *notes* positioned vertically on a *staff*. [22] A practical observation about harmony is the way choirs (not the verse choir kind of choir; the singing kind) operate – the singers are grouped into usually four, and are given different melodies to be sung and when they sing altogether, they create harmony.

Lastly, texture is referred to as the number of individual melodies and the relationship the melodies possess to one other [3]. It is also described as the way the three elements mentioned earlier are combined and blended [22]. There are three kinds of texture in music: monophonic, polyphonic, and homophonic. If the music consists of melodies standing alone without any accompaniment, it is described as *monophonic*; otherwise, if there are two or more independent melodies sounded simultaneously, it is referred to as *polyphonic*. The *homophonic* can somewhat be said as similar to the polyphonic kind, except that in homophony, the other melodies that can be heard support a prominent melody. [3] [22]

Other elements such as dynamics and tone color refer to how relatively loud or relatively soft music is being played, and to the difference in sound quality produced, respectively. Tone color is also referred to as the timbre. Different musical instruments may have different timbres as well as human voices. [3]

1.1.2 Musical Instruments

Musical instruments are special devices created to produce musical sounds [22]. In a traditional orchestra, these instruments are grouped into four families, depending on how they are played and the sound that they produce. These families are the woodwind family, strings family, brass family, and the percussion family.



Taken from [35]

Figure 1.1: Instruments in an Orchestra

Instruments belonging to the woodwind family are those in which wind is blown through it in order to produce sound. Brass instruments are also played in the same manner. The difference between these two classes of instruments is the quality of sound or the timbre that they produce most primarily because of the material they are made of. Brass instruments produce a heavier and a raspier sound whereas woodwind instruments create this soft soothing sort of sound. Examples to these kinds of instruments are the trumpet and the flute, respectively. String instruments, on the other hand, are those played either by plucking the strings or by bowing the strings. Commonly known string instruments are the violin and the double bass. Meanwhile, the instruments belonging to the percussions are played by striking usually with the use of a drumstick or a beater. The timpani and the xylophone are some of the instruments that fall under this category. [21]

In a typical simple band setup, the instruments observably present are the drums, electric or acoustic guitars, keyboards, and bass guitar. In a concert or a gig, the drums usually serve as the band's metronome and it is the main instrument that gives the beat. It goes hand in hand with the bass guitar in maintaining the rhythm of the music being played. The guitars and the keyboards, on the other hand, have the primary role of supporting and harmonizing with the main melody of the music, and in some cases, may

even be given the task of bringing out the melody or tune of the music, if there are no vocals. The bass guitar, as mentioned earlier works together with the drums, and in some cases, may harmonize with the melody. Additionally, the bass guitar is said to establish the foundation of the music being played.



Taken from [32]

Figure 1.2: A Typical Rock Band Setup

1.1.3 Genre

One of the most basic ways humans categorize music is by genre. Through time, it has become a popular approach to organizing and classifying and grouping music [13] [14]. Though there are no universally accepted definitions as to what a genre is, it can be safe to assume that all would agree to the idea that genre means a kind of music, as suggested by Franco Fabbri [18]. Genre often serves as a criteria for choosing and buying music, and it also seems to give listeners a prejudice towards unknown songs as to whether liking it or not. R'n'b, jazz, and rock are among the popular genres people of all walks and ages listen to.

1.2 Bass Line

Most people, whether a Music major or just an ordinary person who loves listening to music, can distinguish a bass line from the other melodies in a song or in a music due to its low-pitched sound. Formally, the bass line is a monophonic instrumental melody usually played by instruments such as the bass guitar and the double bass. In most musical styles, it has the role of bridging together the melodic and the rhythmic sections [28]. It is perhaps the most important part of music as it is said to be “the groundwork or foundation upon which all musical composition is to be erected [2].”

The way bass lines run vary depending on the genre of its piece. For instance, the bass line to a song may consist of constantly repeating notes in a steady rhythm. This is true for rock music. For disco music, we would very much often hear the bass lines playing along octaves, and for reggae music, along thirds and fifths. Whereas for jazz, the bass lines have varying characteristic movements that may run on a syncopated or irregular beat. And because of these facts, we can all agree to the hypothesis that bass lines are sufficiently rich in information useful for accurate genre identification [28].

1.3 Music Data

Like a picture which is said to be figuratively capable of painting a thousand words, music, too, is able to give its listeners rich and meaningful information. There exist three types of music data: audio recording, symbolic musical representation, and cultural data. Audio recordings are physical audio signals captured and represented in a digital manner. Such data are stored in formats like MP3s, WAVs and FLACs. Symbolic musical representations, or symbolic data, on the other hand, are representations of sound based on musically meaningful symbols such as musical notations, information on melody and chords, and many more. The MIDI file format is one of the most commonly used storage format for this kind of data, along with the OSC and Humdrum formats. Meanwhile, cultural data refers to information that is not a direct representation of the actual sounds of the music, but nevertheless relevant to the music at hand, such as its weekly rankings

in MTV, MYX, etc. One will not be lost when trying to get this kind of data for the Internet is a rich repository of a music piece's metadata. [16]

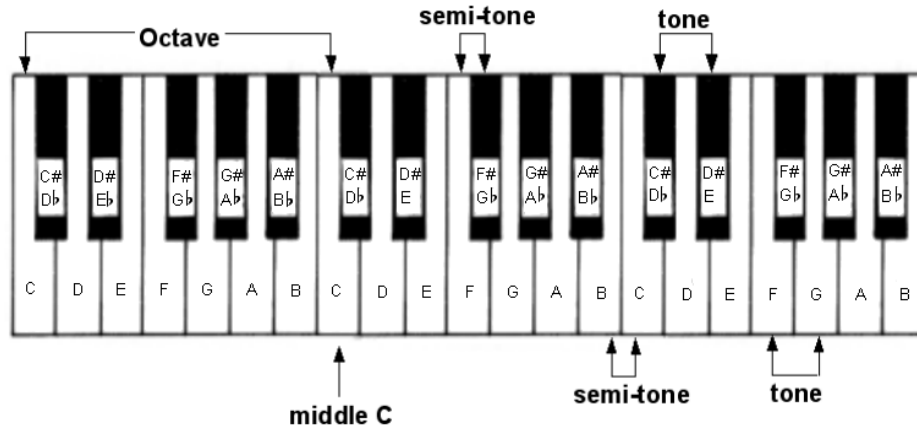
1.3.1 High-level Music Features

Different kinds of features can be extracted from music data. There are features that contain spectral or time-domain information. These kinds of features are called low-level features, extracted directly from the audio signals itself. On the other hand, there are features that consist of information on musical abstractions meaningful to musically trained individuals such as chord frequencies and the instruments present in a musical score. These features are referred to as the high-level features, and can be extracted from symbolic data such as MIDI files. Meanwhile, there are also features that comprise of statistical information such as playlist co-occurrence and purchase correlations. This kind of features is called the cultural features and these features can easily be mined from the Web. [17]

This study focuses on using high-level musical features for the determination of a musical score's genre. In particular, features regarding melody and melodic intervals are the ones employed.

1.3.1.1 Melodic Intervals

A melodic interval is one of the high-level musical features that can be extracted from symbolic data. It represents the number of semitones, or the distance, between two adjacent notes in a given time interval [14]. A semitone is a half-step in a musical scale. In other words, a semitone exists between a note and its corresponding sharp or flat. One octave, or notes from *C (do)* to a higher *C (do)* contains twelve semitones.



Taken from [33]

Figure 1.3: Illustration of Semitones in a Piano

1.4 MIDI Format

MIDI is short for Musical Instrument Digital Interface. It is an encoding system for the representation, transfer, and storage of symbolic musical information. It consists of a series of instructions termed MIDI messages that correspond to an event or change in a control parameter. [14] Many formats exist for symbolic data storage such as OSC and Hundrum, but the MIDI is the more popular one and is best known by the general public. The MIDI was originally created as an interface for instruments coming from different manufacturers to control functions such as note events like note on and off events, timing events, pitch bends, pedal information, program channel information and many more. [9]

The information contained in MIDI files is referred to as a *sequence*. In turn, this sequence is comprised of one or more *tracks*. Each of these tracks typically contains notes being played by a single instrument, like let's say, the drums or the bass guitar. For a MIDI file to play, it needs a *sequencer* – a software or device that is able to read sequences and deliver MIDI messages in its proper timing. The sequencer is synonymous to a conductor in an orchestral setup who tells who plays when. [24]

1.5 Machine Learning

Today, we live in a world where not only humans are required to think, but machines and devices are now expected to think and make basic decisions and such. For it to become possible, machines, like human beings, must have some kind of mechanism for learning. In artificial intelligence, learning refers to some kind of algorithm that seeks to reduce the error on a training set. [31] Among machines, learning happens as it observes its interactions with its environment and its own decision-making processes. It comes in three cases: supervised learning, unsupervised learning, and reinforcement learning [25].

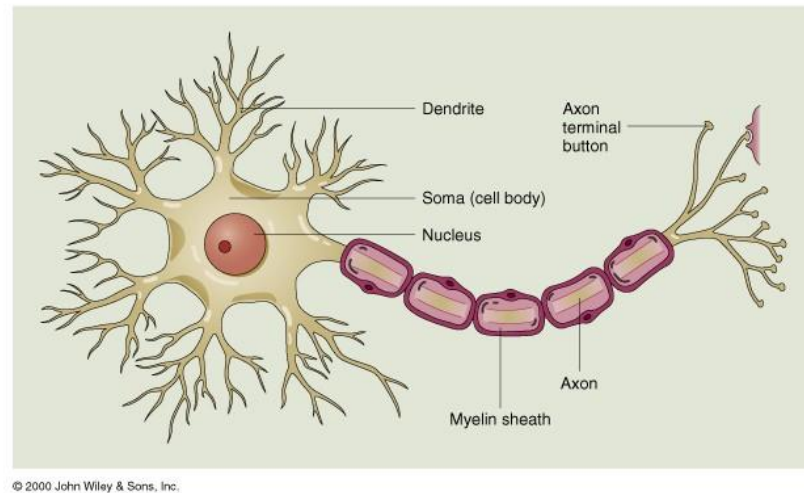
In supervised learning, the agent is presented with examples of possible inputs and its corresponding output, meanwhile in unsupervised learning, the agent is shown patterns about the input but the output values for each are not supplied. In reinforcement learning, on the other hand, the agent learns through the concept of “reinforcement”, or rewards. [25] This particular study will deal with supervised learning, which will classify a given music according to its genre, given a set of training data that consists of inputs (music) and its respective output classification.

1.6 Neural Networks

The neural network is one of the most popular forms of machine learning systems and it is even considered to be among the most effective. Its architecture is inspired by the brain’s information-processing ability and the connections formed by the neurons [25]. Artificial Neural Networks (ANNs) are usually utilized in pattern recognition and data classification problems. [1]

1.6.1 Neurons in the Real World

The neuron is the basic unit of the nervous system. It is a cell that sends messages or impulses to glands, muscles, and other neurons alike. [29]



Taken from [30]

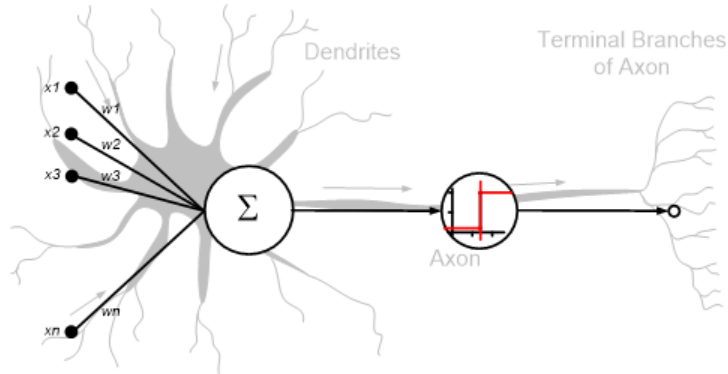
Figure 1.4: The Biological Neuron

There are three basic components of the neuron, namely the axon, dendrite, and cell body. The axon is responsible for the transmission of neural impulses to its neighbouring neurons while the dendrites are responsible for receiving the said impulses or messages. These axons and dendrites are projected from the cell body. [29]

In between the axon of one neuron and the dendrite of another neuron is what we call the synapse. It contains a slight gap where the transfer of signals from one neuron to the next neuron takes place. When a neural impulse travels down to the terminal button of the axon, it triggers the secretion of a neurotransmitter. This neurotransmitter is a chemical that stimulates the next neuron so that an impulse transmission can take effect. Neurotransmitters either have an excitatory effect or an inhibitory effect on neurons, which makes the inside of the receiving neuron more positive than on the outside, or more negative on the inside than on the outside, respectively. When the receiving neuron is depolarized enough by the excitatory neurotransmitter, it fires a neural impulse, thus transmission takes effect. [29]

1.6.2 The Artificial Neuron and ANN

Similar to the biological neurons, the neurons of the artificial neural network perform the same task of transmitting messages to the next neuron. The dendrite, cell body, and axon are represented as the input weights, activation function and the output weight, respectively.



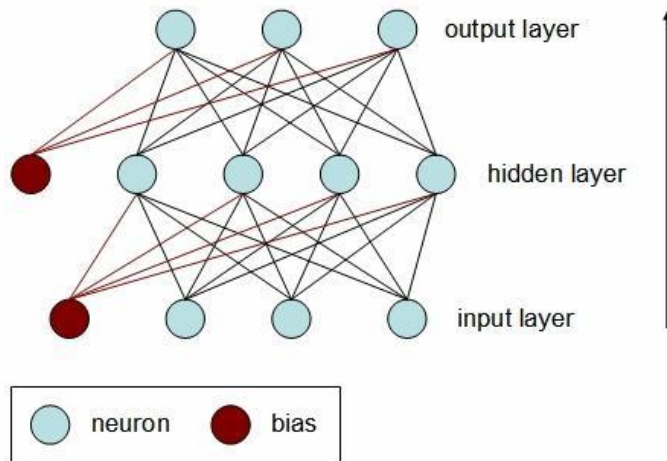
Taken from [36]

Figure 1.5: The Artificial Neuron

The artificial neural network (ANN) can be viewed as an abstraction of the nervous system. Similar to the neural connections of the human body, the ANN is modelled as a collection of nodes connected via directed links. Each link is associated with a numeric weight that represents the strength of connection between the two nodes at its ends. The output of a node is determined by an activation function, which takes on the summation of the node's inputs and its corresponding weights. The most commonly used activation function is the logistic function, more known as the sigmoid function, written mathematically as

$$f(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

where $= \sum(\text{input } x \text{ weight})$. [25]



Taken from [26]

Figure 1.6: Multilayer Perceptron

1.6.3 Network Structures

The neural network comes in different structures. It may be either feed-forward (acyclic) or recurrent (cyclic). Feed-forward networks are a function of its current input, while recurrent networks are a function wherein the output is fed back as inputs. [25]

Feed-forward neural networks also come in different forms. When a feed-forward network consists of only an input layer and an output layer, it is called a single-layer neural network, or a perceptron. This is the simplest form of a neural network in which the inputs are directly connected to the weights. Meanwhile, if a feed-forward network already contains a hidden layer aside from its input and output layers, it is then referred to as a multilayer feed-forward network, or a multilayer perceptron. The addition of a hidden unit enables the network to represent a larger space or scope of hypotheses to the problem it's trying to verify. [25]

1.6.4 Learning and Classification

Neural networks, like humans, learn by example. The neural network is trained using a series of sample inputs, and from there, it learns how to make generalizations. Thus, when presented with an unknown input, the network will be able to classify based on what it has learned. [1] Learning in neural networks is synonymous to the scenario where you teach a child what a dog, a cat and a fish looks like, and the child will be able to identify a dog, whether what he or she sees is a Labrador or a Siberian Husky.

Neural networks are iteratively trained, usually with the use of the backpropagation algorithm, or metaheuristics like the Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Cuckoo Search Algorithm.

1.7 Extreme Learning Machine

The Extreme Learning Machine (ELM) is a recent breakthrough in computational intelligence that tries to overcome the challenges encountered among traditional computing techniques such as the Support Vector Machines (SVMs) and the Neural Networks (NNs). Developed by Huang, et al., the ELM addresses issues such as slow learning speed, over-fitness, local optima entrapment, and improper learning rate, which are very much common in neural networks. Though the ELM was created for single-hidden layer feed forward neural networks, its hidden layer does not need tuning, unlike the traditional NNs. [6]

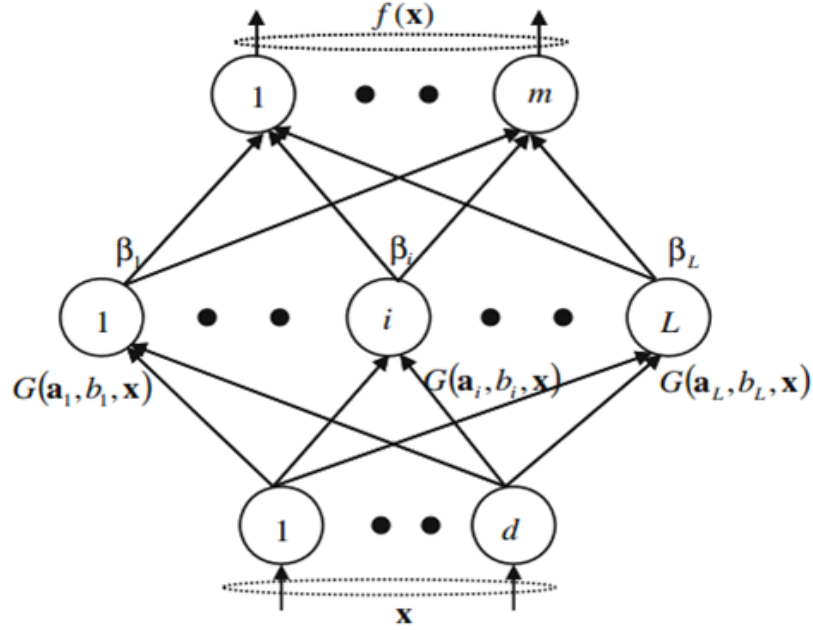


Figure 1.7: Structure of a Single-Hidden Layer Feed-forward Neural Network

1.7.1 Three-step Learning Algorithm

The basic ELM algorithm can be summarized by the following three-step procedure: [7]

ELM Algorithm:

Given a training set $\mathbf{X} = \{(x_i, t_i) | x_i \in \mathbf{R}^d, t_i \in \mathbf{R}^m, i = 1, \dots, N\}$, hidden node output function $G(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x})$, and hidden node number L ,

- step 1* Randomly generate hidden node parameters $(\mathbf{a}_i, \mathbf{b}_i)$, $i = 1, \dots, L$.
- step 2* Calculate the hidden layer output matrix \mathbf{H} .
- step 3* Calculate the output weight vector β :

$$\beta = \mathbf{H}^\dagger \mathbf{T}, \text{ where}$$

$$\mathbf{T} = [t_1, \dots, t_N]^T.$$

First, the hidden node parameters a and b are generated in a random manner. This serves as the input weights linking the input to each of the hidden nodes, and the bias for

each of the hidden nodes, respectively. The hidden node parameters come in the form of matrices having the size $d \times L$ for parameter a , and L for parameter b , where d is the dimension of the input vector and L is the number of hidden nodes.

The next step is the computation of the hidden layer output matrix \mathbf{H} . Its computation is done using the hidden node output function $G(a_i, b_i, x)$ for all items in the training data set. The matrix \mathbf{H} can be written as:

$$\begin{aligned} \mathbf{H} &= \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} \\ &= \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \vdots & \vdots \\ G(a_L, b_L, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L}. \end{aligned} \quad (2)$$

The hidden node output function G represents the activation function. The most commonly used activation is the sigmoid function.

The last step in this three-step process is the calculation of output weight vector β , where

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad (3)$$

with L as the number of hidden nodes, as stated earlier, and m as the number of output nodes. This vector β is derived from $\mathbf{H}\beta = \mathbf{T}$, where

$$\mathbf{T} = \begin{bmatrix} t_1^T \\ \vdots \\ t_L^T \end{bmatrix} \quad (4)$$

Thus, vector β becomes $\mathbf{H}^\dagger \mathbf{T}$. \mathbf{H}^\dagger is the Moore-Penrose generalized inverse of matrix \mathbf{H} , computed in the following manner:

$$\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T, \quad (5)$$

where \mathbf{H}^T is the transpose of matrix \mathbf{H} and $(\mathbf{H}^T \mathbf{H})^{-1}$ is the inverse of the matrix formed by multiplying matrices \mathbf{H}^T and \mathbf{H} .

1.7.2 ELM with Regularization Parameter

In order to increase the stability and the generalization performance of the ELM, a regularization parameter, or a positive value $1/C$ (or λ) is added to the diagonal of $\mathbf{H}^T \mathbf{H}$ or $\mathbf{H} \mathbf{H}^T$ in computing for β [6]. Thus, the formula of the output weight vector β becomes

$$\beta = \mathbf{H}^T \left(\frac{I}{C} + \mathbf{H} \mathbf{H}^T \right)^{-1} \mathbf{T} \quad (6)$$

With this kind of ELM implementation, the classification performance of the ELM is regardless of the dimension of the feature space, or the number of nodes in the hidden layer. Even with a hidden node count of 1000, good generalization results are obtained, for as long as the number of hidden nodes is large enough. [8]

1.7.3 Learning without Iterative Tuning

Unlike the other algorithms used to train networks such as back-propagation and meta-heuristics, ELM learns without the need for an iterative tuning process. Meaning, all the training happens in just one cycle. Even though the parameters are random-generated, ELM is still able to arrive at an optimum solution. Just like in a system of linear equations, the Moore-Penrose generalized inverse of a matrix, let's say, matrix G , is able to find a solution matrix x to the set of linear equations $Ax = y$, such that such that Gy is a minimum norm least-squares solution to the above linear system, and $G = A^\dagger$. [7]

Chapter 2

Literature Review

Genre classification is one of the widely-studied areas in the field of Music Information Retrieval (MIR). Many attempts have been made in solving the music genre classification problem, which involve different approaches and algorithms such as K-Nearest Neighbour, Hidden Markov Models and Support Vector Machines, and features such as Mel-frequency cepstral coefficients and melodic interval histograms.

McKay and Fujinaga, in their paper entitled “Automatic Genre Classification Using Large High-Level Feature Sets,” showed how large sets of high-level musical features can be effective for utilization in the genre classification task. These features were categorized into those based on instrumentation, on texture, on rhythm, on dynamics, on pitch statistics, on melody, and on chords. Additionally, these features can also be grouped as to either one-dimensional or multi-dimensional. MIDI recordings comprised the data set, and the MIDI recordings were first classified according to its root genre, then subsequently according to its corresponding leaf genres, thus the hierarchical classification. One experimental setup used a taxonomy comprising of 3 root genres and 9 leaf genres and another setup made use of a taxonomy consisting 3 root genres and 38 leaf genres. A hybrid classifier of feed-forward Neural Networks (NN) and K-Nearest Neighbor (KNN) was used in coming up with the decision for the input’s genre classification. KNN was applied to the one-dimensional features, while the multi-dimensional features were processed using NN. Genetic Algorithms (GAs) were used all throughout in selecting the features and the weights. Success rates of 98% for the root genres and 90% for the leaf genres were achieved in their setup that employed a taxonomy comprising 3 root genres and 9 leaf genres. [15]

Meanwhile, Karpov performed a study in music genre classification using Hidden Markov models (HMMs) as the underlying algorithm to solve the said classification problem. His data set consisted of digital audio recordings in MP3 format, which were

later on converted to PCM-encoded WAV files. These were then partitioned into ten-second segments. From these partitioned pieces, Karpov extracted distinct types of features namely the Fourier transform-based Mel cepstral coefficients, Mel cepstral with delta and acceleration information, and linear predictive encoding. Separate experimental setups were made for the different feature types mentioned a while ago. Training was done using a continuous multivariate hidden Markov model, with one training model for each genre specified. The experiments showed quite promising results, which can reach up to a 94.4% success rate, depending on the number of states configured in the models, on the feature sets used, and on the number of genres being represented in the data set as well. [11]

Hagglade, et al. also tried solving the automatic music classification problem, and with the additional exploration of image-to-genre mapping. Music files in .au format were used all throughout the research, and features from these music files were extracting using Mel-frequency cepstral coefficients (MFCCs). Different machine learning algorithms such as K-Nearest Neighbour, K-means, multiclass SVM, and neural networks were investigated in classifying music into classical, jazz, metal, and pop genres. The results had an expected favor towards neural networks and SVMs in comparison with the KNN and K-means classifiers. At the same time, the results showed a surprising superiority of neural networks than SVMs. The data set used in their study was of two format types: the mono PCM format and the MP3 format. A multivariate autoregressive (MAR) feature model was then designed and several classifiers were tested such as least squares error-minimized linear model, Gaussian classifiers, Gaussian Mixture Model classifier, and Generalized Linear Model classifier. The results given by the classifiers still went through post-processing methods such as majority voting and sum-rule before arriving at a decision. Experimentations with different setups gave results showing that human classification, with 98% accuracy rate on an average, performed about 18% better than the best performing MAR features proposed approach. [5]

Another attempt in solving the music genre classification problem was made by Meng, et al. In their paper, they focused on the identification of temporal feature

integration methods. In general, temporal feature integration is the process of combining multiple different features in a time window into a single feature vector. It is said that this method is able to capture relevant temporal information needed for genre classification. Meng, et al. extracted short-time features, medium-time features and long-time features from the audio samples. Examples of such features are the Mel-frequency cepstral coefficient (MFCC), the high zero-crossing rate ratio (HZCRR), and the beat histogram feature, respectively. These feature vectors are then partitioned into uniform short-time windows, and then are all combined to form a single new feature vector in a larger time scale. [19]

Simsekli also took on the challenge of creating an automatic music genre classification system. Unlike the previous approaches, Simsekli focused his study on bass lines, believing that bass lines are a rich source of information that can be sufficient for accurate genre identification. In this research, Simsekli used the same MIDI recordings that McKay and Fujinaga used in their study as datasets, but manually discarding all the instrumentation aside from that of the bass line part. Melodic interval histogram features were then extracted from the bass lines. The K-Nearest Neighbor (KNN) classifier was chosen to be the machine learning algorithm for this particular classification task. Similar to McKay and Fujinaga's approach, hierarchical classification using a 3-root-genre-9-leaf-genre taxonomy was employed in this study. Experiment results show that a maximum accuracy rate of 84.44% can be obtained for this genre classification dilemma using KNN classifiers. [28]

In this study, the method proposed by Simsekli served as a major reference, except that the classifier used is the Extreme Learning Machine (ELM) instead of the K-Nearest Neighbor (KNN) classifier.

Many studies have showed the superiority of ELM over the traditional classification methods. One of these studies is that of Loh and Emmanuel. They have showed in their study how ELM can be a good classifier for music genre classification. Their study, "ELM for the Classification of Music Genres," revolved around RIFF-

WAVE audio formats for the data set, with zero crossing rates, energy, root-mean-square, crest factor, spectral centroid, Mel-frequency cepstral coefficients, and specific loudness sensation for features. Loh and Emmanuel's experiments showed that ELM performed better than SVMs, with ELM having a classification accuracy of 85.312%, while SVMs having 82.8125%. [12]

The ELM algorithm was also used in a study that is directly related to the field of medicine and health. In Karpagachelvi, et al.'s paper, they experimented on the classification of Electrocardiogram, or ECG signals using ELM as the classifier. The ECG is a recording of the electrical activity of the heart, and was also used in heart-disease investigations. Results of the experiments on using ELM in classifying five kinds of abnormal waveforms and five kinds of normal beats showed the superiority of the generalization capability of ELM in comparison to the traditional classifier SVM. [10]

The field of bioinformatics was also touched by ELM. One of the areas of research in this particular field is on protein classification. In the research conducted by Wang and Huang entitled "Protein Sequence Classification using Extreme Learning Machine," ELM was used to automatically classify protein sequences based on a given of 10 protein super-types. The sigmoidal function was used as the ELM's activation function and a Gaussian RBF kernel was also employed. In the experimentation phase, ELM was compared with BP-NN, or backpropagation-trained neural networks. Results show that the ELM had a very much less training time and a slightly better performance in classification accuracy than the BP-NN classifier. [34]

ELM also made its way to the power consumption and utility areas of research. Nizar, Dong, and Wang made a research on power utility non-technical loss (NTL) analysis and used ELM to classify patterns on power consumption behaviour. It is said that data coming from consumer electricity consumption behaviour can be used as a way of detecting cases of NTLs in the utility market. In their study, three classification algorithms were used in grouping customer consumption behaviour, namely the ELM, Online-Sequential ELM (OS-ELM) and SVM. Their performances were compared and

the results showed an overall superiority of the two ELMs compared to the SVM. Furthermore, OS-ELM had better classification accuracy than the ordinary ELM [23]. OS-ELM classification is used for cases when the training set needs frequent updating [6].

The ELM algorithm was also used as a classification method for a multicategory cancer diagnosis based on microarray data. In this paper by Zhang, et al., ELM was compared with neural networks and SVM algorithms. The ELM classifier was able to perform the multicategory classification directly, without any modification unlike in SVMs. Moreover, the ELM classifier was able to achieve a higher classification accuracy rate than the rest of the algorithms tested in Zhang, et al.'s research, with less training time and a smaller network structure. [37]

Chapter 3

Statement of the Problem

Music genre classification is one of the widely-studied areas in the field of Music Information Retrieval, along with playlist generation, music recommendation, cover song detection, music fingerprinting, and many more [14]. Genre classification is basically about being able to give a genre label to a particular music based on how it sounds or how it is played.

Different genres give a different feel to the listeners. There are genres that give a relaxing feeling such as bossa nova or smooth jazz, while other genres give a heavy feeling like hard rock and metal. There are cases that when the music gives a bouncy feel and somewhat makes you feel you would want to dance, listeners associate it to pop, when in fact it can be electronica. There are also cases when the music is ridiculously noisy, it is already associated to rock music when in fact there are different kinds of rock music, such as alternative rock, that can give you a mellow feeling. Through this observation alone, the definition of genre can be somewhat problematic.

Normal people with untrained ears in music classify what they hear based on the limited knowledge on music genres they have. Not all people are experts in this field, nor do they have the opportunity to learn every existing genre and how they sound like. Thus, the need for an automatic music genre classification system arises.

The Extreme Learning Machine (ELM) is one of the well-performing machine learning algorithms used for solving classification problems. In this study, ELM will be used as a solution to the music genre classification problems, with a focus on bass lines as the basis for the genre classification.

Chapter 4

Objectives

The objectives of this study are outlined below:

1. To apply the ELM algorithm to music genre classification,
2. To evaluate the performance of ELM in genre classification by varying the number of its hidden nodes,
3. To evaluate the performance of ELM in genre classification by introducing a regularization coefficient, and
4. To evaluate the performance of ELM in genre classification by altering the network structure.

Chapter 5

Proposed Approach

5.1 Overview of the Approach

Figure 5.1 illustrates the flow of the proposed ELM approach to the music genre classification problem.

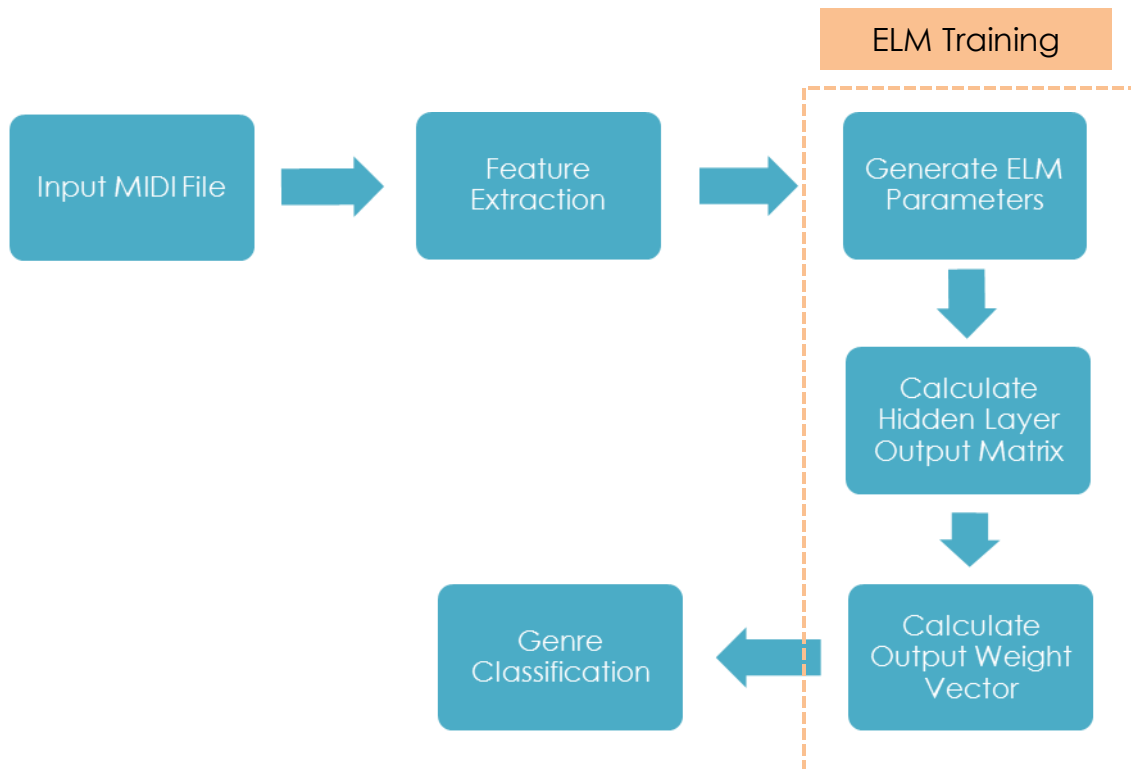


Figure 5.1: Program Flow

5.2 Input

The input is a MIDI sound file that exclusively contains the bass lines to a particular song or musical piece. Any length or duration of MIDI file will do.

5.3 Input Representation and Feature Extraction

The representation of the input MIDI file is a feature vector of length 128 containing the values of its melodic interval histogram. The said histogram is a frequency count of the number of semitones between two adjacent MIDI messages containing pitch values, or note-on MIDI messages. After the extraction of the features from the input, the resulting histogram is normalized. This normalization process is done by dividing the frequency in each bin in the histogram by the total number of observations, so that the magnitude of each bin specifies a fraction of all melodic intervals that match the melodic interval of that particular bin. [14] The length of the feature vector represents the range of possible pitch values contained by note-on MIDI messages, which is from 0 to 127.

In GenrELM, this feature extraction method is handled by *jSymbolic*, an open-source platform for extracting high-level features from MIDI data. *jSymbolic* is under the *jMIR* suite created by McKay and Fujinaga for extracting information useful for research in the Music Information Retrieval area. [14]

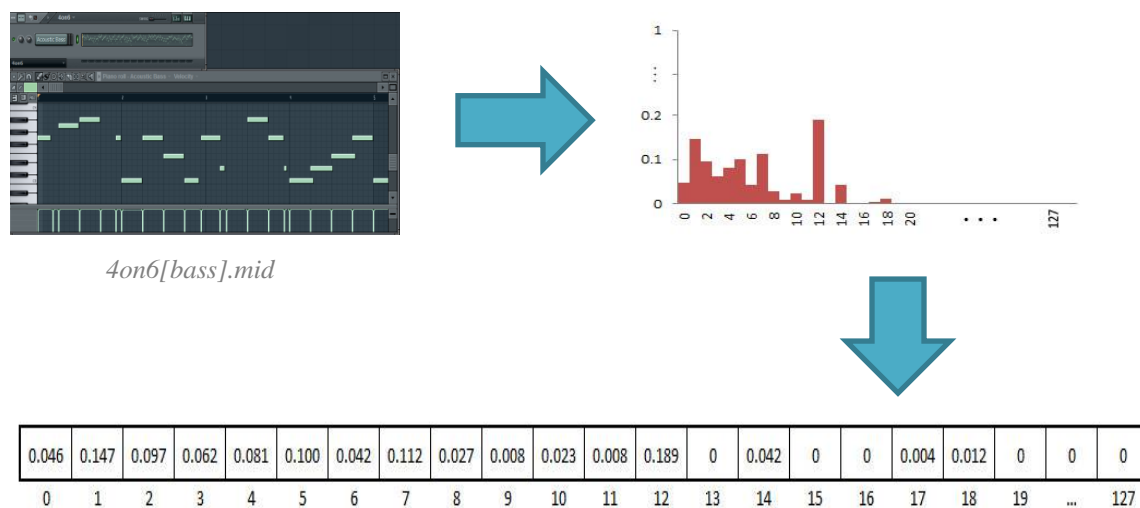


Figure 5.2: Feature Extraction and Input Representation

5.4 Genre Taxonomy

The input files are classified according to root genres, and then are further classified according to the corresponding leaf genres. The table below describes the genre taxonomy that is used as basis for genre classification.

Table 5.1: Genre Taxonomy Adopted in GenrELM

Jazz	Rhythm & Blues	Rock	- root genres
Bebop	Blues Rock	Hard Rock	} leaf genres
Swing	Funk	Metal	
Bossa Nova	Rock ‘n Roll	Alternative Rock	

The above taxonomy was the same genre taxonomy used by Simsekli in a similar study. [28]

5.5 Data Set

The data set consists of 225 MIDI files taken from the data set used by McKay and Fujinaga in their study [15]. Bass lines were then manually extracted from these MIDI files using a third-party music-editing software.

5.6 Training with ELM

5.6.1 Training Set

The training set consists of 80% of the data set, divided equally among the genres. Since there are four ELM classifiers, one for the root genres and three for the leaf genres corresponding to the three root genres, there will also be four sets of training sets. The feature vectors extracted from each set of training set will be contained in a text file.

5.6.2 Network Structure

Figure 5.3 shows the structure of the ELM network to be employed in this study:

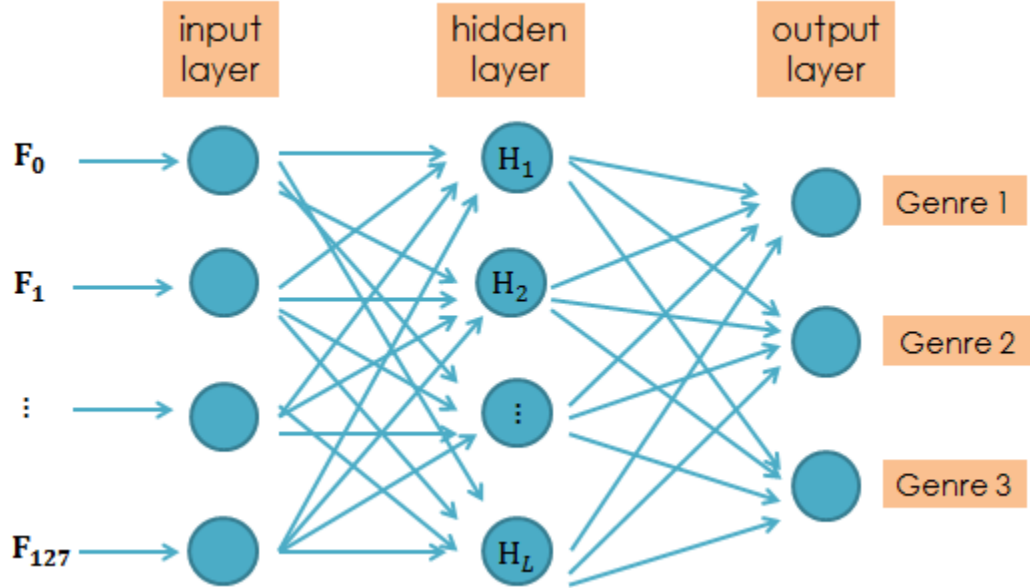


Figure 5.3: The ELM Network Structure for GenrELM

5.6.3 Input Nodes

The number of input nodes corresponds to the length of the feature vector. In this case, the number of input nodes is 128.

5.6.4 Nodes in the Hidden Layer

The number of nodes in the hidden layer is user-defined and is not restricted to the number of training samples to be employed. Thus it is possible to create a network with 1000 hidden nodes regardless of the quantity of the training data.

5.6.5 Output Nodes and Target Output Mapping

The number of output nodes is set to three for both networks handling the root genres and the leaf genres, since there are three root genres and three leaf genres per root genre.

As for the target output mapping, the table below shows the values encoded for each node in order to describe each genre, both roots and leaves:

Table 5.2: Target Output Encodings for Root Genres

Genre	Node 1	Node 2	Node 3
Jazz	1	0	0
Rhythm & Blues	0	1	0
Rock	0	0	1

Table 5.3: Target Output Encodings for Jazz Leaf Genres

Genre	Node 1	Node 2	Node 3
Bebop	1	0	0
Swing	0	1	0
Bossa Nova	0	0	1

Table 5.4: Target Output Encodings for Rhythm ‘n Blues Leaf Genres

Genre	Node 1	Node 2	Node 3
Blues Rock	1	0	0
Funk	0	1	0
Rock ‘n Roll	0	0	1

Table 5.5: Target Output Encodings for Rock Leaf Genres

Genre	Node 1	Node 2	Node 3
Hard Rock	1	0	0
Metal	0	1	0
Alternative Rock	0	0	1

5.6.6 Extreme Learning Machine Training Flow

Figure 5.4 summarizes how the ELM training step is done:



Figure 5.4: ELM Training Flow

The first step is to randomly generate the hidden node parameters a (input weights towards each node in the hidden layer) and b (bias for each node in the hidden layer) with values ranging from -1 to 1.

$$a = \begin{bmatrix} -0.804 & \cdots & -0.855 \\ \vdots & \ddots & \vdots \\ -0.299 & \cdots & -0.181 \end{bmatrix}_{d \times L} \quad \text{input weights}$$
$$b = \begin{bmatrix} 0.528 \\ \vdots \\ -0.269 \end{bmatrix}_L \quad \text{bias}$$

d – input vector length
 L – no. of hidden nodes

Figure 5.5: Hidden Node Parameter Generation

After which, the hidden layer output matrix H is calculated using the random-generated parameters. The activation function used here is the sigmoid function. Then, the output weight vector β is derived using H and the target output vector T (see section 1.7).

Training is done independently for the networks handling the root genres and the leaf genres. In effect, the resulting number of matrices for the input weights a is four – one for the root genres and one each for the leaf genres under the root genres. The same goes for the number of hidden node biases matrices b and output weight vectors β . These resulting matrices are stored in text files for use in the testing step. Unlike the traditional methods for training neural networks, training in ELM happens in one sweep only.

5.7 Testing the Network

In this setup, testing is done in a two-step manner: first, the testing is done for the network handling the root genre classification, and based on the resulting root genre from this network, a second network corresponding the leaf genres to this root genre is now tested.

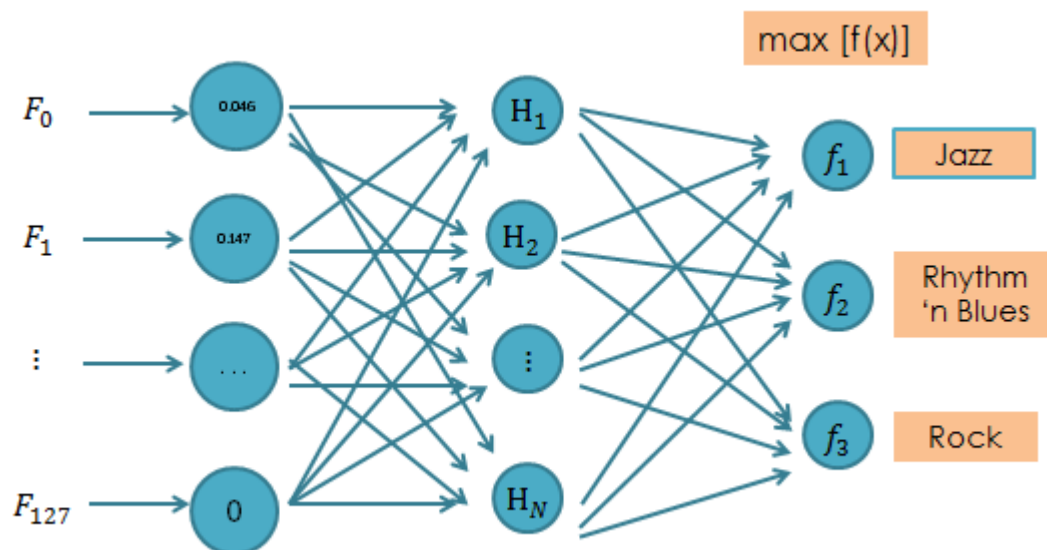


Figure 5.6: Root Genre Classification of *4on6[bass].mid*

Jazz	Rhythm & Blues	Rock	- root genre
Bebop	Blues Rock	Hard Rock	{ leaf genres
Swing	Funk	Metal	
Bossa Nova	Rock 'n Roll	Alternative Rock	

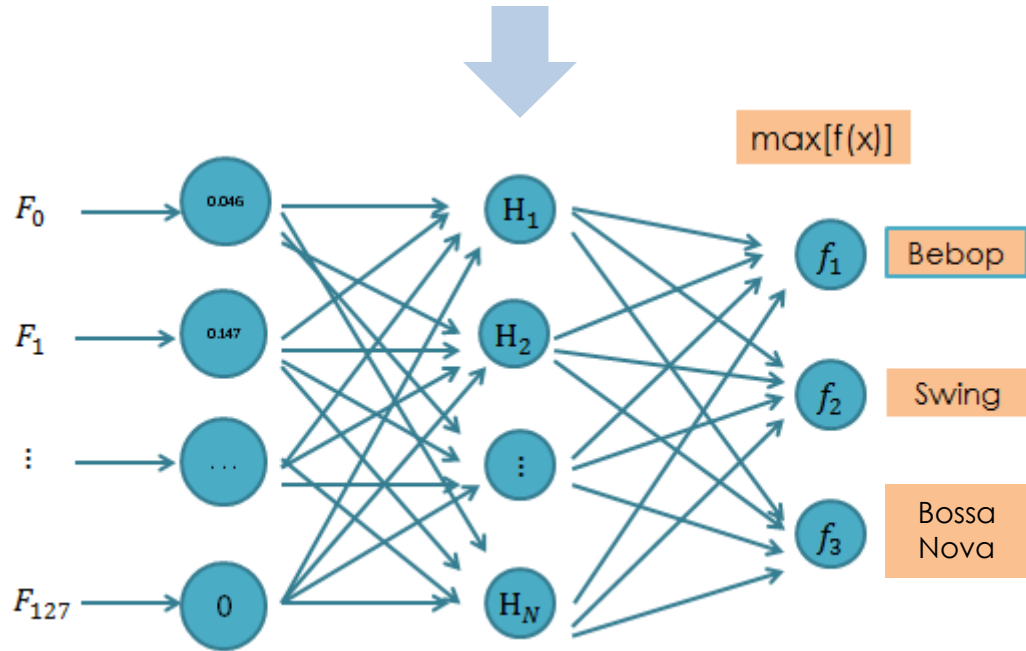


Figure 5.7: Leaf Genre Classification on *4on6[bass].mid*

In testing the network, the hidden node parameter matrices a and b and the output weight vector β are first retrieved from the text files from whence they are stored. The computation for the hidden layer output matrix H is done in a same fashion as what was done in the training phase. After which, the values of this matrix H is multiplied to the output weight vector β and are summed up per output layer node in order to get the final value. Since the music genre classification is a multiclass classification problem, the output node with the maximum value determines the genre classification of the input MIDI music file. The table below indicates the maximum output node-to-genre mapping basis for the classification results:

Table 5.6: Maximum Output Node-to-Genre Mapping for Root Genres

Output Node with Maximum Value	Genre
1	Jazz
2	Rhythm & Blues
3	Rock

Table 5.7: Maximum Output Node-to-Genre Mapping for Jazz Leaf Genres

Output Node with Maximum Value	Genre
1	Bebop
2	Swing
3	Bossa Nova

Table 5.8: Maximum Output Node-to-Genre Mapping for Rhythm ‘n Blues Leaf Genres

Output Node with Maximum Value	Genre
1	Blues Rock
2	Funk
3	Rock ‘n Roll

Table 5.9: Maximum Output Node-to-Genre Mapping for Rock Leaf Genres

Output Node with Maximum Value	Genre
1	Hard Rock
2	Metal
3	Alternative Rock

5.8 Output

The output of the ELM-trained network is a leaf genre classification of the input MIDI bass line file. The input may only fall under one of the following leaf genres: bebop, swing, bossa nova, blues rock, funk, rock 'n roll, hard rock, metal, or alternative rock.

Chapter 6

Experiments and Results

6.1 System Specifications

In presenting the feasibility and effectiveness of ELM in automatic music genre classification, experiments were performed using a machine with the following specifications:

Hardware

Intel Core i5 CPU M 460 @ 2.53 GHz

2048 MB RAM

Software

Microsoft Windows 8 Pro 32-bit (6.2, Build 9200)

Eclipse Juno IDE

Java Virtual Machine

6.2 Experimental Settings

The application program was coded in Java using the Eclipse IDE. All experimental setups were subjected to 30 trials in order to test the feasibility of using the ELM algorithm in music genre classification using bass lines.

As mentioned previously, the dataset consists of 225 MIDI files partitioned into separate sets for training and testing. Partitioning was done on a random-stratified basis. The table below shows how the dataset was divided.

Table 6.1(a): Dataset Partitioning for GenrELM

	Training	Testing
Jazz	60	15
Rhythm and Blues (R'nB)	60	15
Rock	60	15
Root	180	45

Table 6.1(b): Dataset Partitioning for GenrELM (cont'd.)

		Training	Testing
Jazz	Bebop	20	5
	Swing	20	5
	Bossa Nova	20	5
R'nB	Blues Rock	20	5
	Funk	20	5
	Rock 'n Roll	20	5
Rock	Hard Rock	20	5
	Metal	20	5
	Alternative	20	5

The datasets for the Jazz, R'nB, and Rock classifiers make up the dataset for the root-level classifier. The training sets were also tested in order to see how well the ELM algorithm models the training samples.

In most of the experimental setups, the initial parameters for the ELM are defined as follows:

A. Basic ELM (w/o Regularization Coefficient)

No. of hidden nodes 10

B. ELM w/ Regularization Parameter

Regularization coefficient	0.2
----------------------------	-----

6.3 Experimental Results and Discussions

Here are the following experimental setups that were employed in this study:

- Variation in Training and Test Sets Used
- Variation in Number of Hidden Nodes
- Introduction of a Regularization Coefficient
 - Variation in Regularization Coefficient Values
- Variation in Network Structure (i.e. no. of output nodes)
- Comparison with Existing Studies

Though the proposed method is first to classify the input MIDI files according to the root genres, then further classify it based on the leaf genres belonging to the initial classification, the leaf classification on the MIDI files were done as if they were correctly classified in the root level. This is in order to see the maximum potential of the ELM algorithm when applied to music genre classification.

6.3.1 Variation in Training and Test Sets Used

Since the dataset used in this study is not standard in terms of how the MIDI files were partitioned into sets for training and testing, this setup performs training and testing using different sets per run. This is to see the robustness of the ELM algorithm. Different hidden node values were explored, ranging from 10 to 100 by intervals of 10. Additional values of 150, 200, 250 and 300 were also tested in the Root classifier.

Table 6.2: Variation in Training and Test Sets Used (Training Set) – Root Results

	10	20	30	40	50	60	70	80	90	100
Mean	76.06	81.61	84.59	86.65	88.65	90.35	91.67	93.22	93.52	95.19
Median	76.11	81.67	84.72	86.67	88.89	90.28	91.94	92.78	93.89	95
Max.	80.56	85	87.22	88.89	92.22	94.44	95	95.56	96.67	97.22
Min.	72.78	79.44	81.67	83.89	85.56	88.33	89.44	91.11	91.11	93.33
Std. Dev.	2.61	1.5	1.59	1.59	2.15	1.29	1.49	1.27	1.64	1.05
Variance	6.8	2.26	2.53	2.52	4.63	1.67	2.23	1.6	2.69	1.09

	150	200	250	300
Mean	99.39	100	100	100
Median	99.44	100	100	100
Max.	100	100	100	100
Min.	98.33	100	100	100
Std. Dev.	0.55	0	0	0
Variance	0.31	0	0	0

Table 6.3: Variation in Training and Test Sets Used (Test Set) – Root Results

	10	20	30	40	50	60	70	80	90	100
Mean	75.33	74.22	74.37	76.07	75.04	75.04	72	71.04	69.26	66.81
Median	77.78	73.33	73.33	75.56	74.44	75.56	73.33	71.11	70	66.67
Max.	84.44	86.67	84.44	95.56	91.11	86.67	84.44	82.22	84.44	77.78
Min.	64.44	66.67	66.67	68.89	64.44	66.67	57.78	60	60	60
Std. Dev.	5.13	5.43	5.18	5.91	6.57	5.65	7.3	6.08	6.66	5.12
Variance	26.34	29.49	26.81	34.97	43.14	31.91	53.33	36.95	44.3	26.2

	150	200	250	300
Mean	54.52	56.67	66.67	37.78
Median	45.19	45.56	62.22	33.33
Max.	45.63	45.56	57.78	33.33
Min.	49.85	48.89	62.22	35.56
Std. Dev.	54.52	56.67	66.67	37.78
Variance	45.19	45.56	62.22	33.33

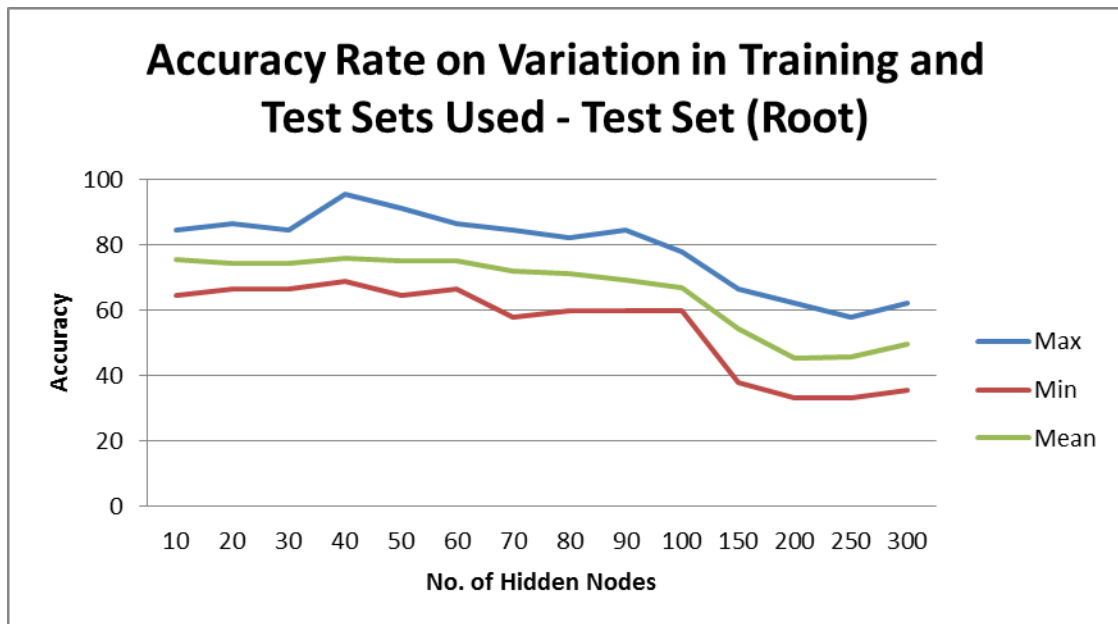
**Figure 6.1:** Accuracy Rate on Variation in Training and Test Sets Used – Test Set (Root)

Table 6.4: Variation in Training and Test Sets Used (Training Set) – Jazz Results

	10	20	30	40	50	60	70	80	90	100
Mean	82.94	89.5	95.22	97.94	99.72	100	100	100	100	100
Median	82.5	90	95	98.33	100	100	100	100	100	100
Max.	90	96.67	100	100	100	100	100	100	100	100
Min.	76.67	85	90	95	98.33	100	100	100	100	100
Std. Dev.	3.71	3.69	2.54	1.56	0.63	0	0	0	0	0
Variance	13.73	13.63	6.46	2.43	0.4	0	0	0	0	0

Table 6.5: Variation in Training and Test Sets Used (Test Set) – Jazz Results

	10	20	30	40	50	60	70	80	90	100
Mean	74.67	76.44	72.67	72.67	59.56	41.56	53.78	55.78	56.22	57.33
Median	73.33	80	73.33	73.33	60	40	53.33	53.33	53.33	60
Max.	100	93.33	93.33	93.33	80	66.67	73.33	80	86.67	80
Min.	60	53.33	53.33	53.33	46.67	26.67	40	33.33	33.33	40
Std. Dev.	10.12	11.44	10.11	10.56	10.35	12.34	10.78	14.28	13.18	10
Variance	102.38	130.98	102.22	111.42	107.08	152.29	116.27	203.78	173.74	99.92

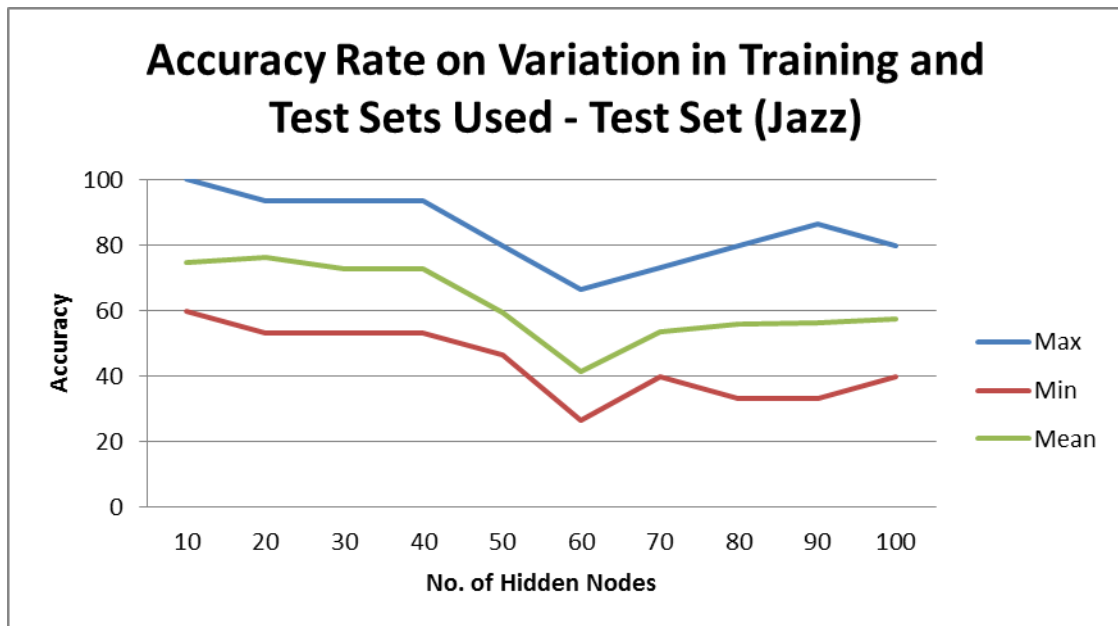
**Figure 6.2:** Accuracy Rate on Variation in Training and Test Sets Used – Test Set (Jazz)

Table 6.6: Variation in Training and Test Sets Used (Training Set) – R'nB Results

	10	20	30	40	50	60	70	80	90	100
Mean	65.17	78.61	89.17	95.61	99.17	100	100	100	100	100
Median	65	78.33	90	95	100	100	100	100	100	100
Max.	75	90	95	100	100	100	100	100	100	100
Min.	56.67	71.67	83.33	91.67	96.67	100	100	100	100	100
Std. Dev.	5.31	4.93	3.49	2.17	1.22	0	0	0	0	0
Variance	28.23	24.35	12.21	4.69	1.48	0	0	0	0	0

Table 6.7: Variation in Training and Test Sets Used (Test Set) – R'nB Results

	10	20	30	40	50	60	70	80	90	100
Mean	47.11	54.67	50.22	48.89	42.67	37.78	40.67	40	42.44	41.78
Median	46.67	53.33	53.33	50	46.67	36.67	40	40	40	40
Max.	66.67	93.33	66.67	66.67	73.33	66.67	73.33	60	66.67	73.33
Min.	20	33.33	33.33	26.67	13.33	13.33	20	20	26.27	20
Std. Dev.	12.25	14.79	11.44	12.17	16.2	13.37	12.91	11.35	9.82	12.74
Variance	149.99	218.85	130.98	148.15	262.38	178.8	166.59	128.74	96.5	162.25

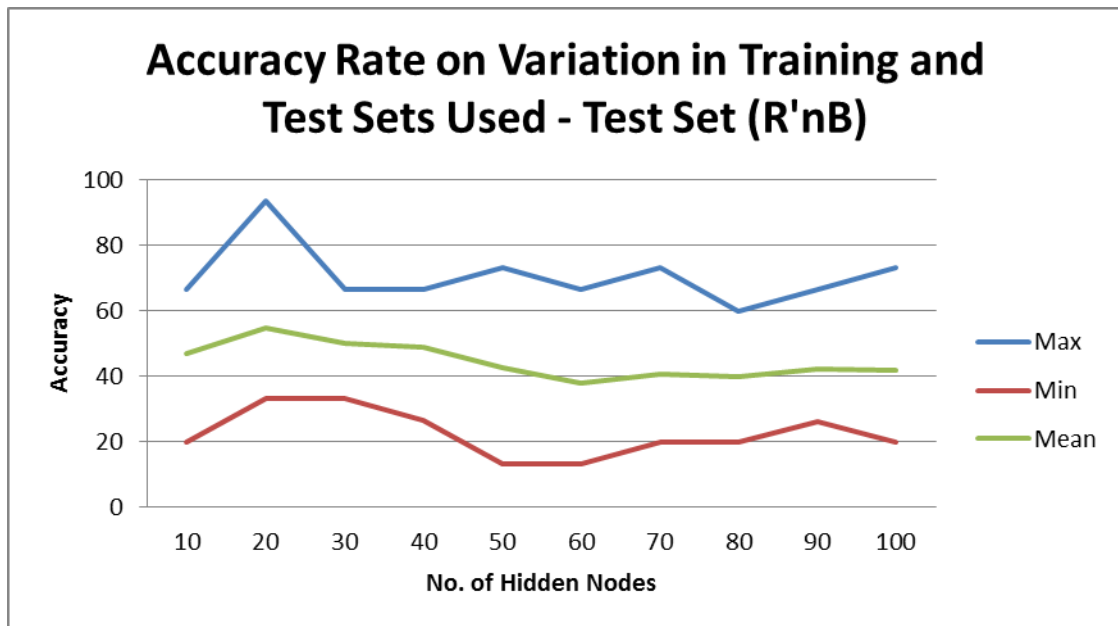
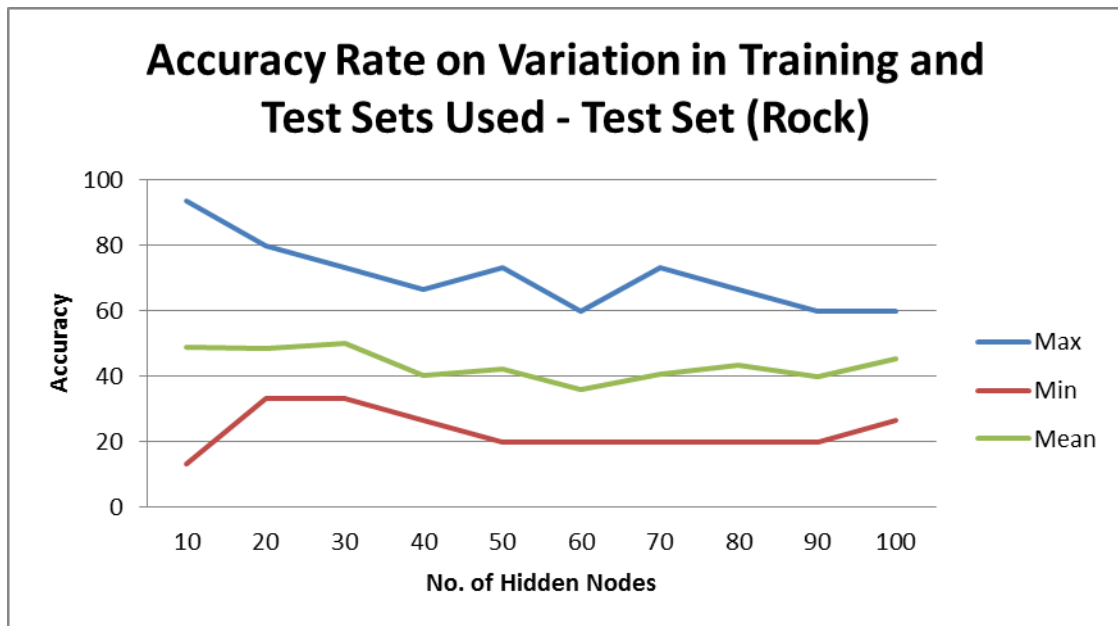
**Figure 6.3:** Accuracy Rate on Variation in Training and Test Sets Used – Test Set (R'nB)

Table 6.8: Variation in Training and Test Sets Used (Training Set) – Rock Results

	10	20	30	40	50	60	70	80	90	100
Mean	64.5	75.72	85.61	92.78	97.94	100	100	100	100	100
Median	65	76.67	86.67	93.33	98.33	100	100	100	100	100
Max.	73.33	85	91.67	96.67	100	100	100	100	100	100
Min.	55	68.33	80	88.33	95	100	100	100	100	100
Std. Dev.	5.5	4.54	3.54	2.41	1.56	0	0	0	0	0
Variance	30.3	20.63	12.54	5.81	2.43	0	0	0	0	0

Table 6.9: Variation in Training and Test Sets Used (Test Set) – Rock Results

	10	20	30	40	50	60	70	80	90	100
Mean	48.89	48.67	50.22	40.44	42.22	36	40.67	43.33	40	45.56
Median	46.67	46.67	53.33	40	40	33.33	40	43.33	40	46.67
Max.	93.33	80	73.33	66.67	73.33	60	73.33	66.67	60	60
Min.	13.33	33.33	33.33	26.67	20	20	20	20	20	26.67
Std. Dev.	14.68	14.24	11.58	11.06	13.02	9.53	12.55	10.9	10.79	10.8
Variance	215.58	202.76	134.05	122.4	169.6	90.73	157.39	118.77	116.48	116.73

**Figure 6.4:** Accuracy Rate on Variation in Training and Test Sets Used – Test Set (Rock)

Figures 6.1 through 6.4 show that for the test sets, maximum accuracy rates of 95.56%, 100%, 93.33% and 93.33% was achieved by the ELM-trained Root, Jazz, R'nB, and Rock classifiers, respectively. As for the training sets, it can be observed, as shown in Tables 6.2, 6.4, 6.6, and 6.8 that when the number of hidden nodes is set to a value greater than or equal to the training set size, a 100% accuracy rate for the training sets was achieved by all the ELM-trained classifiers. This observation is in line with a theorem in ELM which states that if the number of hidden nodes is equal to the number of items in the training set, the training error can be zero [6]. The high accuracy rates obtained for the test sets reflect that the ELM algorithm is robust and can be used in music genre classification.

As what was mentioned previously, the dataset used in this study consists of 225 MIDI files, wherein 80% is for training and 20% is for testing – 60 MIDI files were used for training the leaf (Jazz, R'nB, and Rock) classifiers and 15 MIDI files were tested, while for the root classifier, a combination of the three leaf classifiers. This is quite a small dataset, which serves as the reason behind two observations. One observation is that even though a maximum of 100% was obtained, the minimum accuracy can go as low as 13.33%. Since the training and test sets were varied for each run, there can be instances where the MIDI files that were quite difficult to be classified went to the set for testing and the easier ones went to the training set. Simply put, the ELM classifier was not trained well enough to handle more difficult sets since the knowledge of the classifiers were limited to those MIDI files that are less complicated than the other samples. Another observation is that maximum accuracy rates for the test sets of the R'nB and Rock classifiers is 93.33%. This may seem not quite close from perfect, but out of the MIDI samples tested, only one was misclassified.

6.3.2 Variation in Number of Hidden Nodes

In order to determine the maximum accuracy rate the ELM-trained classifiers can get, different hidden node values were explored using fixed datasets for training and testing. Also, to achieve the highest possible accuracy rate, the dataset was partitioned wherein

there would be a 100% accuracy rate in the root level. The tests were performed wherein the number of nodes in the hidden layer was set to a value falling within the range of 10 and 100, inclusive, by intervals of 10. This setup was done on each on the three classifiers, namely the Jazz, Rhythm and Blues (R'nB), and the Rock classifiers. For the root classifier, the number of hidden nodes explored was extended to values of 150, 200, 250 and 300. The basic ELM, that is, ELM without the regularization coefficient, was adopted in this setup.

Table 6.10: Variation in Number of Hidden Nodes (Training Set) – Root Results

	10	20	30	40	50	60	70	80	90	100
Mean	73.2	77.11	80.91	82.8	85.78	87.69	89.72	91.31	92.78	94.35
Median	73.89	77.22	81.11	82.5	85.83	88.83	90	91.39	92.78	94.44
Max.	76.11	79.44	84.44	85.56	88.89	90.56	92.78	95	95.56	96.11
Min.	68.33	73.33	77.22	80.56	82.78	83.33	86.11	88.33	91.11	92.78
Std. Dev.	1.94	1.35	1.65	1.5	1.81	1.9	1.62	1.55	1.03	0.91
Variance	3.76	1.82	2.73	2.25	3.27	3.62	2.63	2.39	1.06	0.83

	150	200	250	300
Mean	99.15	100	100	100
Median	99.17	100	100	100
Max.	100	100	100	100
Min.	98.33	100	100	100
Std. Dev.	0.6	0	0	0
Variance	0.36	0	0	0

Table 6.11: Variation in Number of Hidden Nodes (Test Set) – Root Results

	10	20	30	40	50	60	70	80	90	100
Mean	86.96	93.11	93.41	90.07	87.33	86	84.89	82.22	75.41	73.7
Median	88.89	93.33	93.33	91.11	86.67	85.56	86.67	82.22	75.56	74.44
Max.	95.56	97.78	100	95.56	95.56	93.33	93.33	88.89	86.67	82.22
Min.	77.78	86.67	86.67	80	77.78	77.78	66.67	73.33	62.22	60
Std. Dev.	4.91	3.11	3.98	4.55	3.74	3.7	5.17	4.21	4.98	5.66
Variance	24.09	9.66	15.83	20.68	14.01	13.67	26.7	17.71	24.84	32.04

	150	200	250	300
Mean	50.96	58.44	58.67	55.78
Median	52.22	60	60	53.3
Max.	62.22	73.33	73.33	73.33
Min.	35.56	46.67	46.67	46.67
Std. Dev.	6.78	8.15	8.65	7.73
Variance	45.95	66.46	74.49	59.72

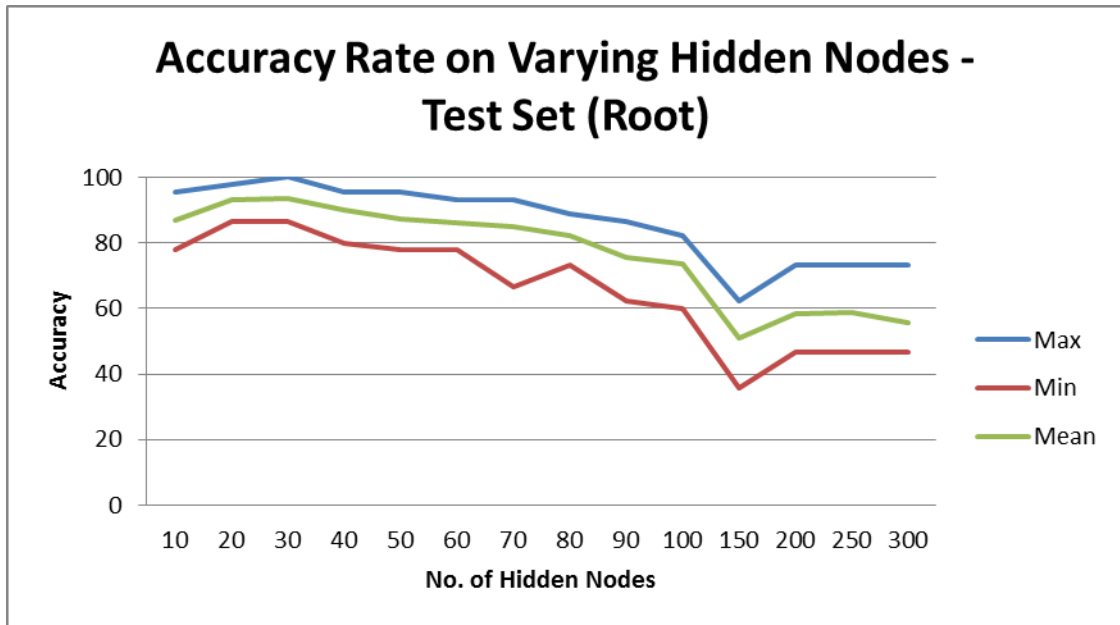
**Figure 6.5:** Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Root)

Table 6.12: Variation on Number of Hidden Nodes (Training Set) – Jazz Results

	10	20	30	40	50	60	70	80	90	100
Mean	79.28	87.28	94.72	97.17	99.44	100	100	100	100	100
Median	79.17	86.67	95	96.67	100	100	100	100	100	100
Max.	88.33	95	98.33	98.33	100	100	100	100	100	100
Min.	73.33	80	90	91.67	96.67	100	100	100	100	100
Std. Dev.	3.52	3.32	2.56	1.46	1.01	0	0	0	0	0
Variance	12.39	11.01	6.53	2.14	1.02	0	0	0	0	0

Table 6.13: Variation on Number of Hidden Nodes (Test Set) – Jazz Results

	10	20	30	40	50	60	70	80	90	100
Mean	92.89	91.56	79.11	73.11	64.89	44	51.56	51.78	57.78	55.33
Median	93.33	93.33	80	73.33	66.67	46.67	53.33	53.33	60	60
Max.	100	100	93.33	100	86.67	66.67	73.33	73.33	80	73.33
Min.	80	80	53.33	53.33	40	26.67	20	40	40	33.33
Std. Dev.	6.99	6.99	11.58	13.5	10.92	11.95	13.89	10.46	10.41	10.81
Variance	48.84	48.84	134.05	182.32	119.34	142.84	192.9	109.37	108.3	116.93

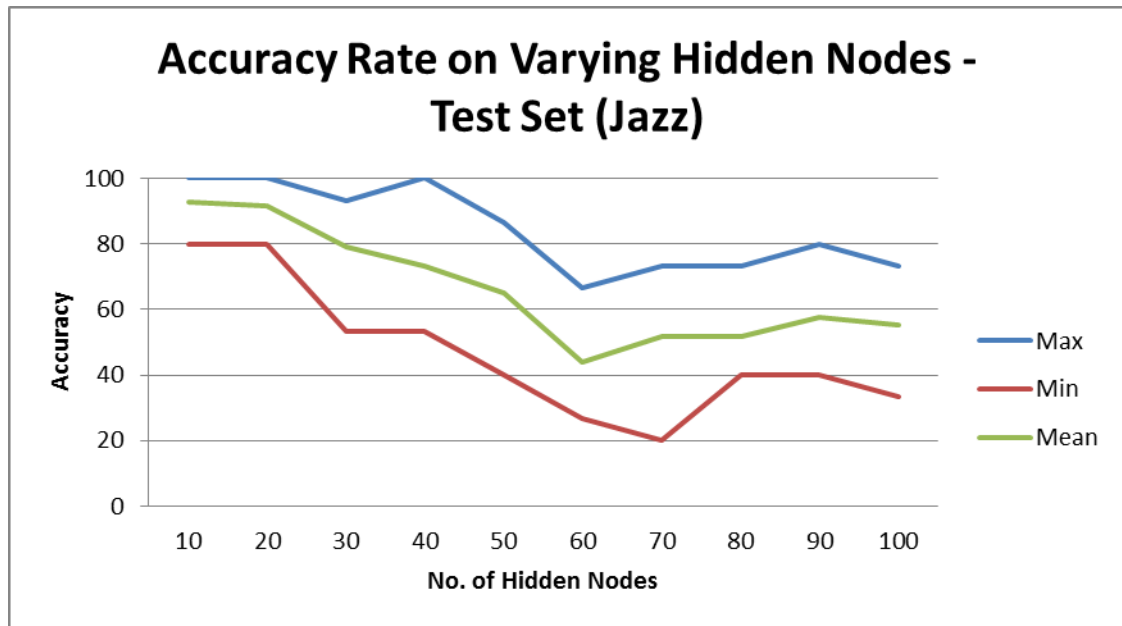
**Figure 6.6:** Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Jazz)

Table 6.14: Variation in Number of Hidden Nodes (Training Set) – R'nB Results

	10	20	30	40	50	60	70	80	90	100
Mean	60.61	74.78	85.83	93.44	98.61	100	100	100	100	100
Median	60	75.83	85.83	93.33	98.33	100	100	100	100	100
Max.	66.67	81.67	91.67	96.67	100	100	100	100	100	100
Min.	55	65	80	86.67	96.67	100	100	100	100	100
Std. Dev.	3.85	4.56	3.02	2.39	1.08	0	0	0	0	0
Variance	14.84	20.83	9.15	5.73	1.17	0	0	0	0	0

Table 6.15: Variation in Number of Hidden Nodes (Test Set) – R'nB Results

	10	20	30	40	50	60	70	80	90	100
Mean	58.22	70.44	60	44.22	38.89	35.56	40.67	37.11	42.22	40.22
Median	60	73.33	60	46.67	40	33.33	40	36.67	40	40
Max.	80	93.33	80	60	66.67	60	53.33	60	66.67	66.67
Min.	40	53.33	33.33	26.67	6.67	13.33	26.67	20	20	20
Std. Dev.	10.05	11.96	10.79	8.66	14.23	13.14	9.8	9.7	12.42	11.28
Variance	100.95	143.09	116.48	75.04	202.55	172.67	96.09	94.05	154.28	127.15

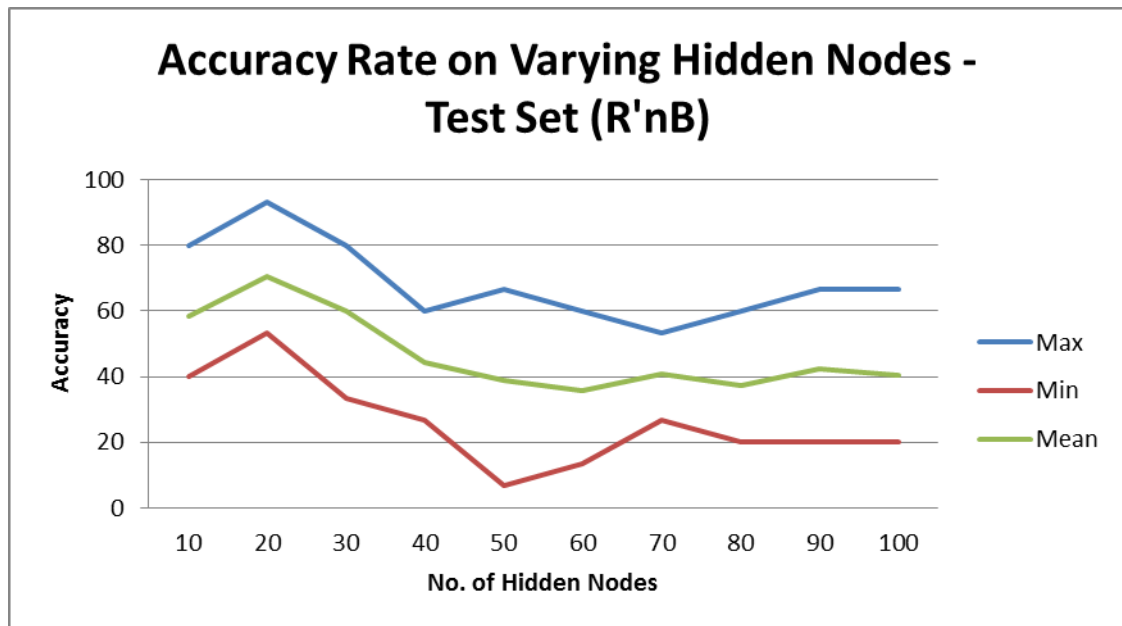
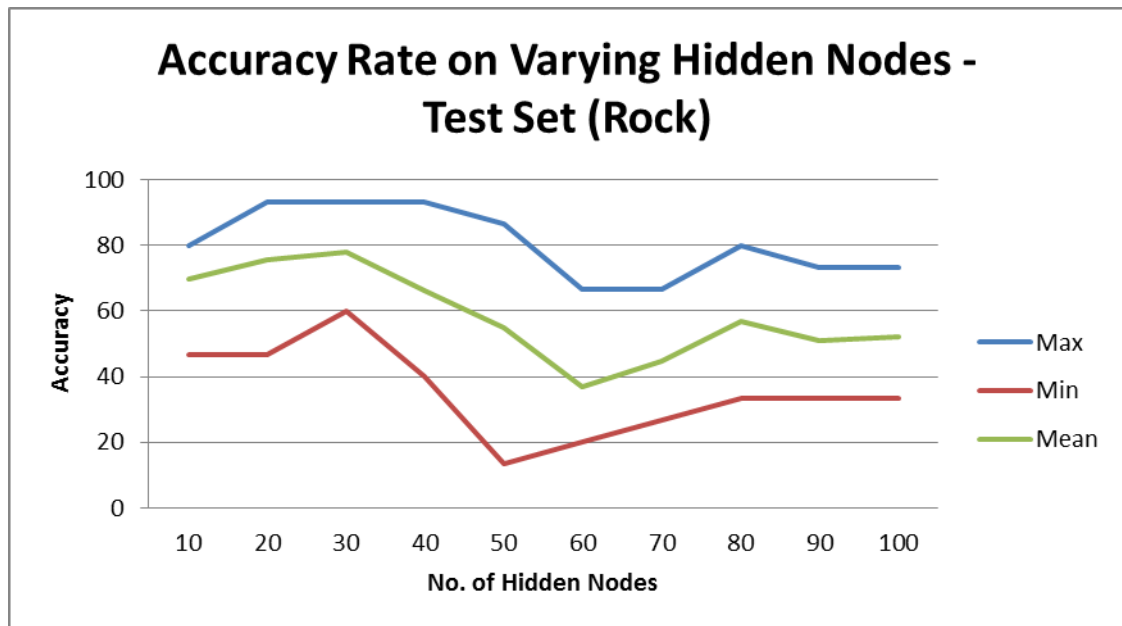
**Figure 6.7:** Accuracy Rate on Varying Number of Hidden Nodes – Test Set (R'nB)

Table 6.16: Variation in Number of Hidden Nodes (Training Set) – Rock Results

	10	20	30	40	50	60	70	80	90	100
Mean	62.17	69.61	80.11	89.11	97.94	100	100	100	100	100
Median	61.67	70	80	90	98.33	100	100	100	100	100
Max.	68.33	75	86.67	96.67	100	100	100	100	100	100
Min.	55	65	75	83.33	95	100	100	100	100	100
Std. Dev.	3.61	2.72	3.03	2.46	1.36	0	0	0	0	0
Variance	13.06	7.41	9.18	6.07	1.86	0	0	0	0	0

Table 6.17: Variation in Number of Hidden Nodes (Test Set) – Rock Results

	10	20	30	40	50	60	70	80	90	100
Mean	69.56	75.56	77.78	66.22	54.67	36.89	44.67	56.67	51.11	52
Median	66.67	80	76.67	66.67	53.33	36.67	46.67	56.67	53.33	50
Max.	80	93.33	93.33	93.33	86.67	66.67	66.67	80	73.33	73.33
Min.	46.67	46.67	60	40	13.33	20	26.67	33.33	33.33	33.33
Std. Dev.	7.96	9.64	9.32	12.37	16.08	10.9	11.89	13.65	10.7	9.97
Variance	63.4	92.98	86.85	153.05	258.7	118.72	141.46	186.21	114.43	99.31

**Figure 6.8:** Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Rock)

The results for the test set, as reflected in Figures 6.5, 6.6, 6.7, and 6.8, show that the basic ELM, that is, without the regularization coefficient, is able to reach a maximum performance of 100% on the Root and Jazz classifiers. Meanwhile, a maximum of 93.33% was attained on the R'nB and Rock classifiers, meaning out of 15 the MIDI files that were tested using this classifier, one was misclassified. Overall, the accuracy rates achieved by the basic ELM in this genre classification study are 100% on the root genres, and 95.56% on the leaf genres. As for the training sets, trends similar with the results of the previous setup were observed as shown in Tables 6.10, 6.12, 6.14, and 6.16, that when the number of hidden nodes is set to a value greater than or equal to the number of items in the training set, the training error is zero.

As mentioned earlier, the number of hidden nodes in the Root classifier was extended to values of 150, 200, 250 and 300, which is up to just a little over its training set size of 180, since the range of the hidden node values for the leaf classifiers were also a little over the size of its training set.

As what can be observed from Figures 6.5, 6.6, 6.7, 6.8, as the number of hidden nodes is further increased, the performance of ELM starts to decline. This trend was also observed in [27] where Shi and Lu stated that the ELM's performance becomes worse when the number of hidden nodes used is too few or too many. The reason behind this is that the ELM becomes a singular least-squares problem that gives an unstable solution. This is especially when the number of hidden nodes is greater than the size of the training set, and when some hidden nodes are assigned with parameters that are correlated to each other. [27]

It can also be seen that the Jazz classifier was able to reach 100% with a hidden node count fewer than the rest of the classifiers. This is because of the nature of the Jazz dataset, wherein the MIDI files can easily be distinguished from each other, even by humans.

6.3.3 Introduction of a Regularization Coefficient

Though in Figure 6.5, the classifiers using the basic ELM can reach a maximum accuracy of 100%, Figures 6.5 to 6.8 show that its average performance lies somewhere between 70% and 90% only. Thus, a regularization coefficient is introduced to the ELM to evaluate whether the overall performance of ELM in genre classification would improve. Furthermore, the tests in this setup were performed with a variation of values for the regularization coefficient (C). The number of nodes in the hidden layer was set to a constant value of 100 for the leaf classifiers and 300 for the root classifier.

Table 6.18: Variation in Regularization Coefficient (Training Set) – Root Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	71.91	72.91	74.37	75.43	75.76	75.43	76.06	77	77.44	77.94
Median	71.67	72.78	74.44	75.56	75.56	75.56	76.11	76.94	77.22	77.78
Max.	72.22	74.44	75.56	76.67	76.67	76.11	77.22	77.78	78.89	78.89
Min.	71.67	71.67	72.22	73.89	75	74.44	75	76.11	76.67	76.67
Std. Dev.	0.28	0.69	0.74	0.66	0.49	0.32	0.64	0.43	0.5	0.57
Variance	0.28	0.48	0.55	0.44	0.24	0.1	0.41	0.18	0.25	0.32

Table 6.19: Variation in Regularization Coefficient (Test Set) – Root Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	85.33	90.3	92.74	93.19	93.33	95.56	95.56	95.7	96.15	97.11
Median	84.44	91.11	93.33	93.33	93.33	95.56	95.56	95.56	95.56	97.78
Max.	88.89	93.33	93.33	93.33	93.33	95.56	95.56	97.78	97.78	100
Min.	84.44	86.67	91.11	91.11	93.33	95.56	95.56	95.56	95.56	95.56
Std. Dev.	1.38	1.98	1	0.56	0	0	0	0.56	1	1.67
Variance	1.91	3.91	1	0.32	0	0	0	0.32	1	2.78

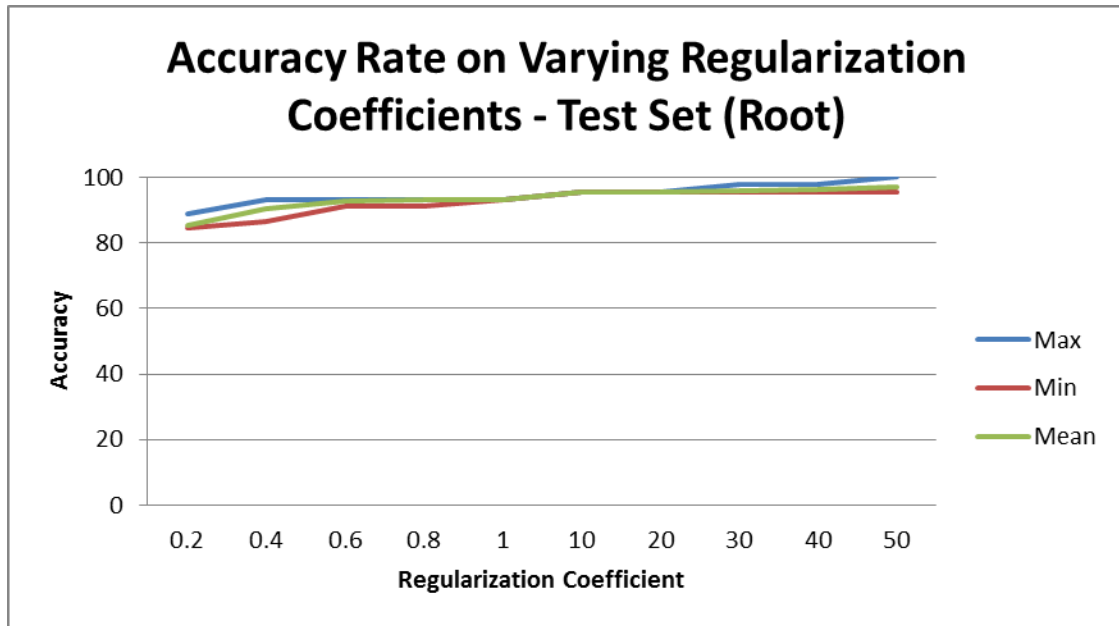


Figure 6.9: Accuracy Rate on Varying Regularization Coefficients – Test Set (Root)

Table 6.20: Variation on Regularization Coefficient (Training Set) – Jazz Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	74.78	75.56	77.44	78.39	78.56	82.78	82.72	83.06	83.33	84.39
Median	75	76.67	78.33	78.33	78.33	83.33	82.5	83.33	83.33	85
Max.	80	80	80	80	83.33	86.67	85	86.67	85	88.33
Min.	70	73.33	73.33	76.67	75	80	80	80	80	81.67
Std. Dev.	2.35	1.57	1.5	1.2	1.68	1.54	1.35	1.58	1.16	1.48
Variance	5.5	2.48	2.25	1.43	2.83	2.36	1.82	2.51	1.34	2.2

Table 6.21: Variation on Regularization Coefficient (Test Set) – Jazz Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	70.67	72.22	73.56	74.22	75.11	99.56	99.78	100	100	100
Median	73.33	73.33	73.33	73.33	73.33	100	100	100	100	100
Max.	80	80	80	86.67	80	100	100	100	100	100
Min.	66.67	66.67	66.67	66.67	73.33	93.33	93.33	100	100	100
Std. Dev.	4.14	3.95	2.76	3.81	3	1.69	1.22	0	0	0
Variance	17.16	15.58	7.61	14.51	8.99	2.86	1.48	0	0	0

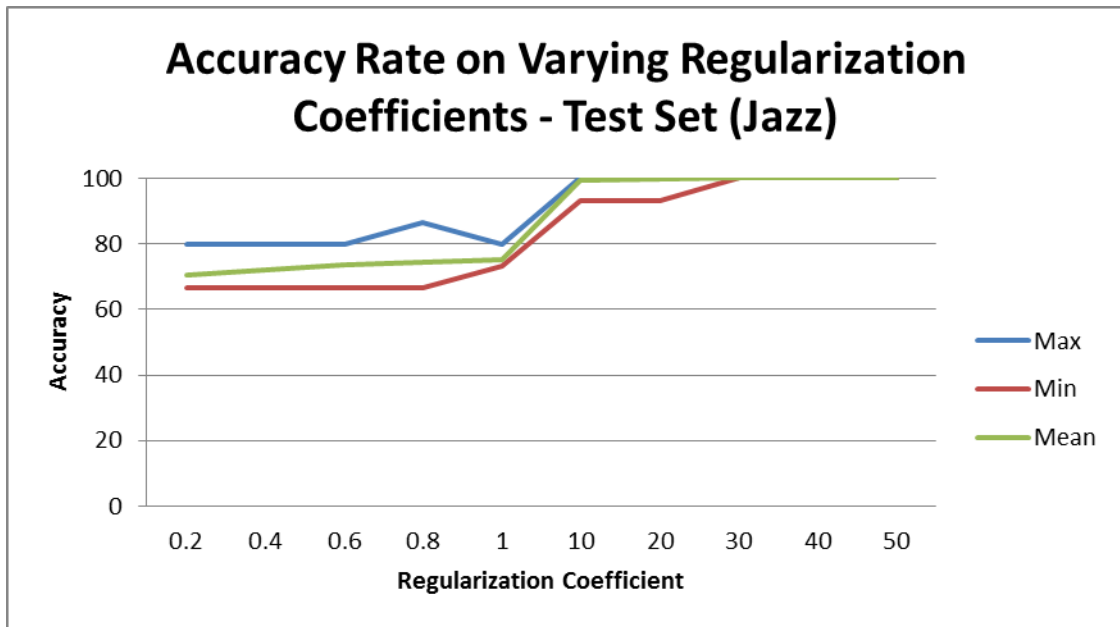
**Figure 6.10:** Accuracy Rate on Varying Regularization Coefficients – Test Set (Jazz)

Table 6.22: Variation in Regularization Coefficient (Training Set) – R'nB Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	51.11	51.5	52.11	53.72	54.22	61.5	66.67	66.94	67.67	67.5
Median	50	51.67	51.67	53.33	55	61.67	65	66.67	67.5	68.33
Max.	56.67	56.67	56.67	58.33	58.33	66.67	70	70	70	70
Min.	48.33	48.33	48.33	48.33	50	58.33	61.67	65	65	63.33
Std. Dev.	1.77	1.97	2.43	2.34	2.47	1.87	1.94	1.58	1.43	2.09
Variance	3.13	3.9	5.93	5.49	6.08	3.52	3.75	2.51	2.03	4.36

Table 6.23: Variation in Regularization Coefficient (Test Set) – R'nB Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	50	49.33	52.22	51.11	52	65.78	68.67	68	67.33	66.22
Median	53.33	46.67	53.33	46.67	53.33	66.67	66.67	66.67	66.67	66.67
Max.	53.33	53.33	60	60	60	73.33	73.33	73.33	73.33	73.33
Min.	46.67	46.67	46.67	46.67	46.67	60	60	60	60	60
Std. Dev.	3.38	3.32	4.32	5.35	5.07	4.87	4.68	4.07	3.2	3
Variance	11.44	11.03	18.65	28.61	25.75	23.7	21.92	16.55	10.27	8.99

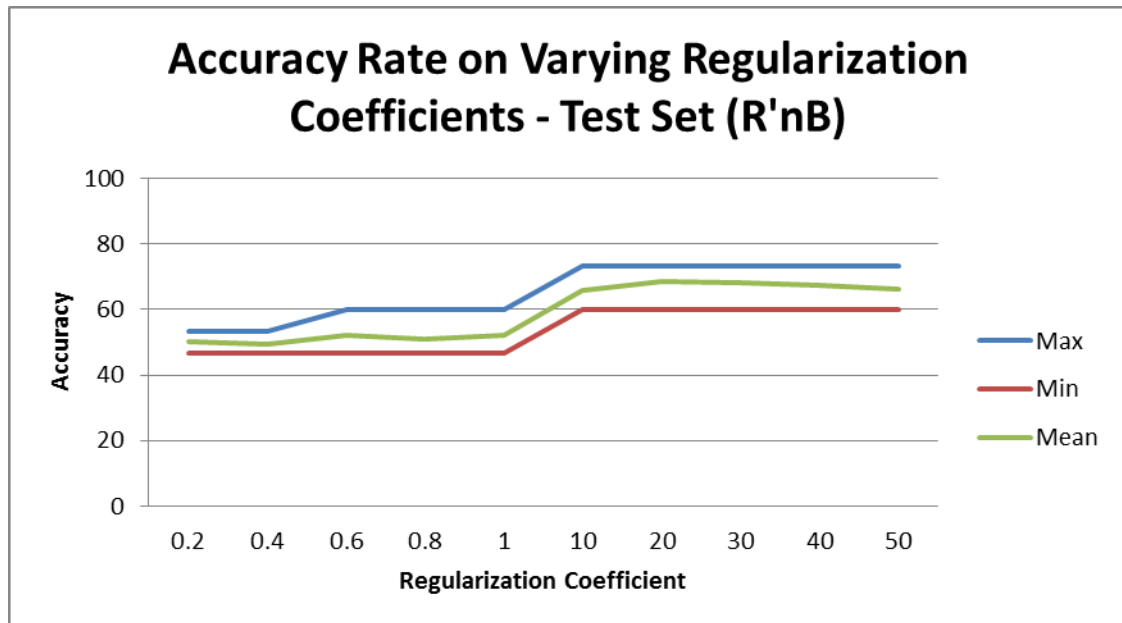
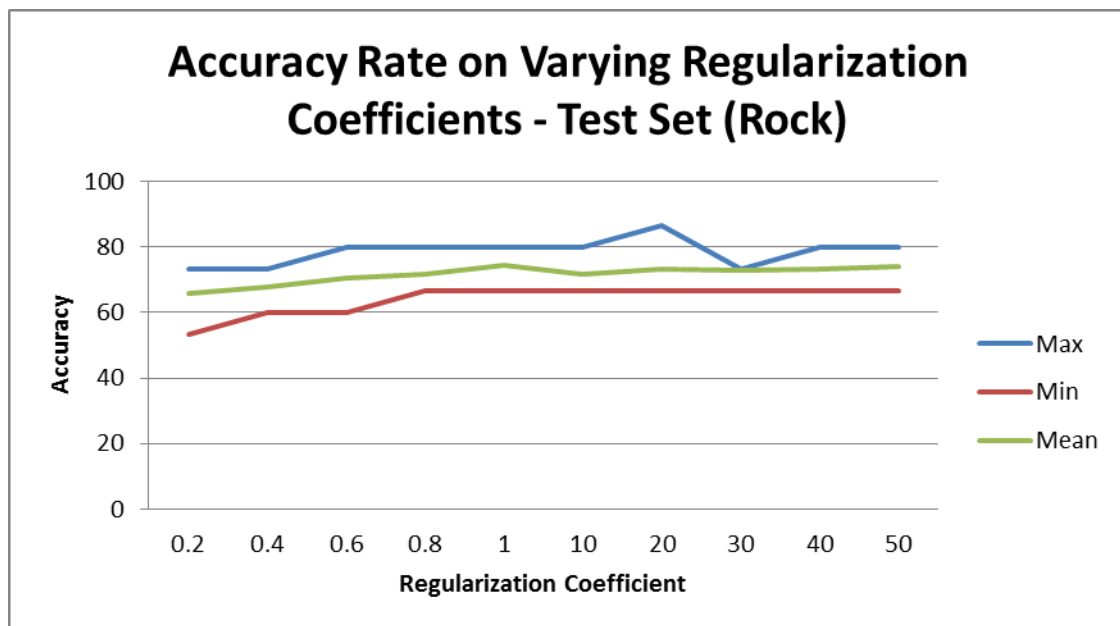
**Figure 6.11:** Accuracy Rate on Varying Regularization Coefficients – Test Set (R'nB)

Table 6.24: Variation on Regularization Coefficient (Training Set) – Rock Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	53.22	55.67	56.22	58.39	57.78	66.83	64.33	62.83	62.89	62.89
Median	53.33	55	56.67	59.17	58.33	66.67	63.33	63.33	63.33	63.33
Max.	56.67	58.33	61.67	63.33	63.33	71.67	68.33	65	66.67	66.67
Min.	45	53.33	51.67	51.67	51.67	61.67	60	60	60	58.33
Std. Dev.	2.59	1.43	2.8	2.98	2.98	2.16	2.03	1.25	1.63	1.69
Variance	6.69	2.03	7.84	8.9	8.88	4.66	4.14	1.56	2.67	2.86

Table 6.25: Variation on Regularization Coefficient (Test Set) – Rock Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	65.78	67.78	70.44	71.78	74.44	71.56	73.11	72.67	73.11	74
Median	66.67	66.67	70	73.33	73.33	73.33	73.33	73.33	73.33	73.33
Max.	73.33	73.33	80	80	80	80	86.67	73.33	80	80
Min.	53.33	60	60	66.67	66.67	66.67	66.67	66.67	66.67	66.67
Std. Dev.	4.19	3.54	4.85	3.79	3.07	3.47	3.27	2.03	2.13	3.2
Variance	17.57	12.52	23.55	14.36	9.45	12.06	10.68	4.14	4.55	10.27

**Figure 6.12:** Accuracy Rate on Varying Regularization Coefficients – Test Set (Rock)

The regularization coefficient is any positive number that when introduced to the ELM classifier, it is able to give more stable solutions [6][27]. As what Figures 6.9 to 6.12 has shown, the maximum, minimum and average accuracies lie close to each other and tend to converge at some point, thus making the overall performance better. Table 6.26 shows a summary of the comparison between the ELM-trained classifier with and without the regularization coefficient in terms of average performance.

Table 6.26: Basic ELM vs. ELM w/ Reg. Coeff – Average Accuracy

	Root		Leaf	
	(300 hidden nodes)		(100 hidden nodes)	
	Training Set	Test Set	Training Set	Test Set
Basic ELM	100%	55.58%	100%	49.18%
ELM w/ reg. coeff ($C. @ 0.2$)	71.91%	85.33%	59.70%	62.15%

The possible maximum accuracy rate that can be achieved by the ELM-trained classifier with a regularization parameter was not anymore explored. This setup was just to show that the ELM with a regularization coefficient performs better than the basic ELM. If the number of hidden nodes is further increased, it is possible to obtain a better performance, for as long as the number of hidden nodes is large enough regardless of the training set size. [8]

6.3.4 Variation in Network Structure

In this setup, the ELM network was restructured into having 9 output nodes. Here, a flat classification will be done, wherein the input MIDI files will be classified directly according to the *genres* *Bebop*, *Swing*, *Bossa Nova*, *Blues Rock*, *Funk*, *Rhythm 'n Blues*, *Hard Rock*, *Metal*, and *Alternative Rock*, which is in contrast to the two-step classification method stated in the proposed approach. This is to see whether there is a significant change in the performance of ELM when the network structure is altered.

Table 6.27: Variation in Number of Hidden Nodes (Training Set) – Flat Classification Results

	10	20	30	40	50	60	70	80	90	100
Mean	50.83	61.35	67.93	73.46	77.85	82.11	84.91	87.59	90.28	92.09
Median	50.83	61.11	68.33	73.61	77.78	82.22	84.17	87.5	90.56	91.94
Max.	55.56	66.67	71.11	78.89	83.33	86.11	88.89	90.56	94.44	95
Min.	47.22	57.22	64.44	70	73.89	78.88	81.11	85	87.78	9
Std. Dev.	2.3	2.4	1.94	2.52	2.28	2.15	2.11	1.33	1.57	1.47
Variance	5.29	6.95	3.74	6.36	5.19	4.61	4.47	1.77	2.46	2.16

	150	200	250	300
Mean	99.09	100	100	100
Median	99.17	100	100	100
Max.	100	100	100	100
Min.	98.33	100	100	100
Std. Dev.	0.52	0	0	0
Variance	0.27	0	0	0

Table 6.28: Variation in Number of Hidden Nodes (Test Set) – Flat Classification Results

	10	20	30	40	50	60	70	80	90	100
Mean	42.81	47.33	47.41	49.04	48.15	50	48.44	47.56	44.89	44.44
Median	44.44	46.67	45.56	48.89	46.67	51.11	48.89	46.67	45.56	44.44
Max.	53.33	60	73.33	62.22	62.22	64.44	60	62.22	53.33	60
Min.	33.33	40	40	37.78	37.78	37.78	37.78	37.78	35.56	37.78
Std. Dev.	6.2	6.15	7.2	6.31	7.23	6.6	6.74	6.77	6.59	5.66
Variance	38.46	37.85	51.88	39.82	52.22	43.51	45.43	45.84	43.39	32.01

	150	200	250	300
Mean	29.33	17.11	20.67	24.89
Median	30	17.78	21.11	24.44
Max.	37.78	24.44	33.33	40
Min.	20	6.67	11.11	15.56
Std. Dev.	5.82	5.39	4.89	6.48
Variance	33.85	29	23.89	42.03

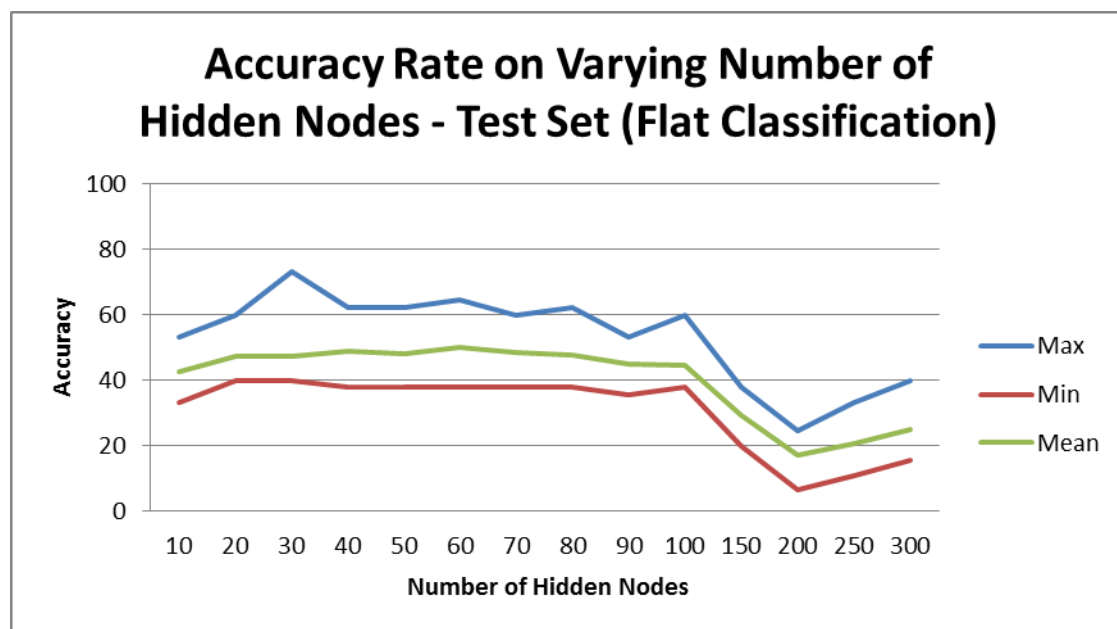
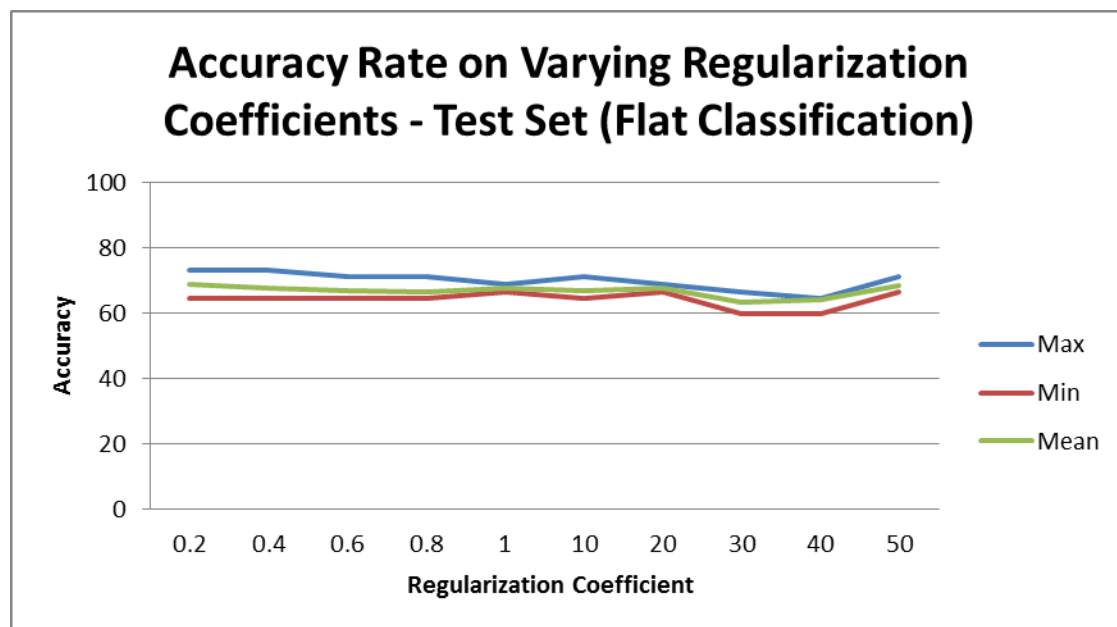
**Figure 6.13:** Accuracy Rate on Varying Number of Hidden Nodes – Test Set (Flat Classification)

Table 6.29: Variation in Regularization Coefficient (Training Set) – Flat Classification Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	47.91	46.09	46.74	47.2	47.96	54.06	55.33	55.89	57.37	58.54
Median	47.78	46.11	46.94	47.22	48.06	53.89	55.56	56.11	57.22	58.33
Max.	50	48.33	47.78	48.89	49.44	55.56	56.67	57.22	60	61.67
Min.	45	44.44	44.44	45.56	46.67	53.33	54.44	53.33	55.5	56.11
Std. Dev.	0.95	0.94	0.96	0.88	0.7	0.6	0.65	0.87	0.97	1.08
Variance	0.91	0.88	0.93	0.78	0.5	0.37	0.42	0.76	0.94	1.16

Table 6.30: Variation in Regularization Coefficient (Test Set) – Flat Classification Results

	0.2	0.4	0.6	0.8	1	10	20	30	40	50
Mean	68.89	67.78	66.74	66.56	67.78	66.89	67.85	63.56	64	68.59
Median	68.89	67.78	66.67	66.67	67.78	66.67	68.89	64.44	64.44	68.89
Max.	73.33	73.33	71.11	71.11	68.89	71.11	68.89	66.67	64.44	71.11
Min.	64.44	64.44	64.44	64.44	66.67	64.44	66.67	60	60	66.67
Std. Dev.	2.33	2.24	2.14	1.5	1.13	1.78	1.13	1.38	1.08	0.96
Variance	5.45	5.02	4.59	2.25	1.28	3.18	1.27	1.91	1.16	0.93

**Figure 6.14:** Accuracy Rate on Varying Regularization Coefficients – Test Set (Flat Classification)

One of the risks posed by hierarchical classification is that when the input MIDI files are misclassified in the root level, the leaf classification is automatically wrong, thus flat classification is investigated in this study. However, as Figures 6.4 to 6.12 is compared with Figures 6.13 and 6.14, hierarchical classification shows a better performance than flat classification, with flat classification attaining only a maximum of 73.33% whereas hierarchical classification had a maximum of 95.56%. This is because there contain similarities between some of the features of the MIDI files under the nine leaf genres. Hierarchical classification was not performed anymore in this setup, since the results stated in the previous pages had shown a 100% accuracy rate for the root level and around 96% for the leaf level.

6.3.5 Comparison with Existing Study

The study on automatic music genre classification using bass lines was performed previously by Simsekli [28] using the K-Nearest Neighbor (KNN) classifier. The KNN classifier is a simple machine learning algorithm that calculates the distance of the given input from its k-neighbors.

Table 6.31 shows a comparison of the results of the test set in this study to that of the previous study:

Table 6.31: KNN vs. ELM Genre Classification Results

Algorithm	Distance Metric / Type	Root Accuracy (%)	Leaf Accuracy (%)
KNN	ED	97.78	84.44
	EMD	93.33	80.00
	KL ₂ D	93.33	80.00
	NCD	75.56	66.67
	PWED	100	86.67
ELM	---	100	95.56

Since the training and test sets used in this study is not standard, thus a direct comparison with the KNN classifier cannot be obtained. However, the results of this study show that a better accuracy rate can be achieved in the music genre classification problem when the classifiers are trained by ELM.

Chapter 7

Conclusion and Future Works

The Extreme Learning Machine (ELM) is a machine learning algorithm that has been applied many times in various optimization problems, and it was shown how ELM can outperform other machine learning algorithms not only in terms of learning speed, but in terms of performance as well. In this study, ELM was used in solving the music genre classification problem.

In this study, the researcher has conducted experiments with results showing how ELM can be a good algorithm for music genre classification. A dataset similar with that of some previous researchers was tested in the ELM classifiers namely the Root, Jazz, Rhythm & Blues, and Rock classifiers. A taxonomy consisting of 3 root genres and 9 leaf genres was adopted by the researcher in this study. In the first and second experimental setups, the leaf accuracies were obtained with the assumption that the MIDI files were correctly classified in the root level.

In the first experimental setup, the training and test sets used for each run was varied. The number of hidden nodes was also varied, with values ranging from 10 to 100, by intervals of 10 for the leaf-level classifiers, while the number of hidden nodes was further extended to values of 150, 200, 250 and 300. Results show an accuracy rate that can go as high as 100%, depending on the training and test sets. This shows that the ELM is robust and is a good algorithm for music genre classification.

In the second experimental setup, the number of nodes in the hidden layers was varied, with values ranging from 10 to 100, by intervals of 10 for the leaf-level classifiers, while the number of hidden nodes was further extended to values of 150, 200, 250 and 300. The results showed that when the number of hidden nodes was further increased, the performance of the classifiers started to decline. Therefore, the ELM algorithm is able to make good generalizations even with few a hidden nodes; however,

once ELM has reached its optimum performance, increasing the number of hidden nodes may not be a wise choice for at some point the overall performance would start to decline.

In the third experimental setup, a regularization parameter was introduced to the ELM. The number of nodes in the hidden layer was set to 100 for the tests involving the leaf-level classifiers, and 300 for the root-level classifier. The results of the tests showed that the maximum, minimum, and average accuracies lie close to each other and tend to converge at some point. The regularization parameter therefore made the ELM more stable. Moreover, the classifiers trained using the ELM with regularization coefficients were made capable of attaining equal accuracy rates throughout the 30 trials.

In the fourth experimental setup, a flat classification approach was implemented, and this was to compare the performance of ELM in music genre classification when the network structure, particularly the number of output nodes, was modified. The results showed a decline in overall performance of the classifier, but the properties of the ELM such as the convergence of the maximum, minimum and average accuracies were preserved. Therefore, in music genre classification, it is more suitable to use the hierarchical approach, even if there is a risk of having an erroneous classification in the root level, thus resulting to an incorrect leaf classification.

In this study, the researcher made use of the basic ELM, that is, without the regularization coefficient, and also investigated the ELM with an added regularization parameter. The researcher recommends for future study that the Online-Sequential ELM (OS-ELM) be applied, so that there won't be a need to retrain the whole dataset if the dataset is updated. The researcher also suggests that the dataset be increased, so as to see if the average performance of ELM will increase. Also, the researcher recommends that the future researcher focus on bass line transcription, which handles extracting bass lines from MP3 files into MIDI bass line files, since it is infeasible and impossible to have the MIDI counterpart for all the existing songs in the world.

References

- [1] An Introduction to Neural Networks. From <http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html> (Accessed: December, 2012)
- [2] Bassline. From <http://en.wikipedia.org/wiki/Bassline> (Accessed: December, 2012)
- [3] Chapter 1 – Elements of Music. From <http://www.wmich.edu/mus-genet/mus150/Ch1-elements.pdf> (Accessed: December, 2012)
- [4] Elements of Music. From <http://www.smccd.net/accounts/mecklerd/MUS250/elements.htm> (Accessed: December, 2012)
- [5] Haggblade, M., Hong, Y., and Kao, K.: Music Genre Classification. From <http://cs229.stanford.edu/proj2011/HaggbladeHongKao-MusicGenreClassification.pdf> (Accessed: December 2012)
- [6] Huang, G.B., Wang, D.H., and Lan, Y.: Extreme Learning Machines: A Survey. *International Journal Machine Learning & Cybernetics*. (2011) 2:107 – 122
- [7] Huang, G.B., Zhu Q.Y., and Siew, C.K.: Extreme Learning Machine: Theory and Applications. *Neurocomputing*, (2006) 70:3056 – 3062
- [8] Huang, G.B., Zhou, H., Ding, X., and Zhang, R.: Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Transactions on Cybernetics*. Vol. 42 No. 2. (2012) 513 – 529
- [9] Introduction to Computer Music: Volume One - Chapter Three: MIDI –. From http://www.indiana.edu/~emusic/etext/MIDI/chapter3_MIDI.shtml (Accessed: December, 2012)
- [10] Karpagachelvi, S., Arthanari, M., and Sivakumar, M.: Classification of ECG Signals Using Extreme Learning Machine. *Computer and Information Science*, Vol. 4, No.1 (2011) 42 – 52
- [11] Karpov, I.: Hidden Markov Classification for Musical Genres. From <http://www.cs.utexas.edu/~ikarpov/Rice/comp540/finalreport.pdf> (Accessed: November, 2012)
- [12] Loh, Q.J.B., and Emmanuel, S.: ELM for the Classification of Music Genres. *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision*, (2006), 1-6

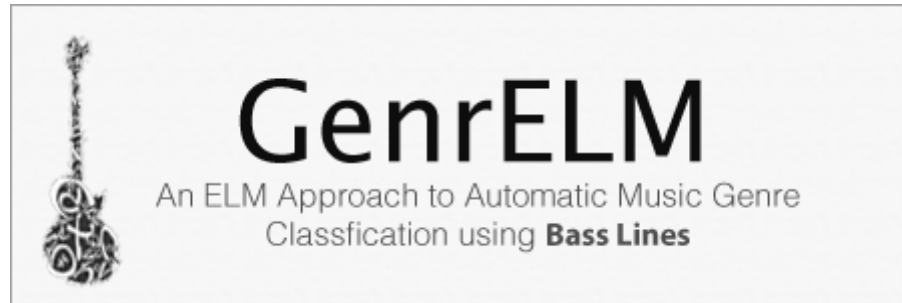
- [13] McKay, C.: Automatic Genre Classification of MIDI Recordings. M.A. Thesis. McGill University, Canada. (2004)
- [14] McKay, C.: Automatic Music Classification with jMIR. Ph.D. Thesis. McGill University, Canada. (2010)
- [15] McKay, C. and Fujinaga, I.: Automatic Genre Classification using Large High-Level Musical Feature Sets. Proceedings of the International Conference on Music Information Retrieval, (2004) 525 – 530
- [16] McKay, C. and Fujinaga, I.: Improving Automatic Music Classification Performance by Extracting Features from Different Types of Data. Proceedings of the ACM SIGMM International Conference on Multimedia Information Retrieval. (2010) 257 – 266
- [17] McKay, C. and Fujinaga, I.: jSymbolic: A Feature Extractor for MIDI Files. Proceedings of the International Computer Music Conference. (2006) 302 – 205
- [18] McKay, C., and Fujinaga, I.: Musical Genre Classification: Is it Worth Pursuing and How Can it be Improved? Proceedings of the 7th International Conference on Music Information Retrieval, Victoria, Canada, (2006)
- [19] Meng, et al.: Temporal Feature Integration for Music Genre Classification. IEEE Transactions on Audio, Speech, and Language Processing, Vol. 15, No.5, (2007) 1654 – 1664
- [20] Music – Merriam Webster Definition. From, <http://www.merriam-webster.com/dictionary/music> (Accessed: December, 2012)
- [21] Musical Instruments – Musical Theory and Technique. From <http://www.thissideofsanity.com/lesson/instruments/instruments.html> (Accessed: December, 2012)
- [22] New Standard Encyclopedia Vol 11, Ferguson Publishing Company, Chicago, Illinois, U.S.A. (1999). M-621 – M-624, M-637
- [23] Nizar, A.H., Dong, Z.Y., and Wang, Y.: Power Utility Nontechnical Loss Analysis with Extreme Learning Machine Method. IEEE Transactions on Power Systems, Vol. 23, No.3, (2008) 946 – 955
- [24] Overview of the MIDI Package. From <http://docs.oracle.com/javase/tutorial/sound/overview-MIDI.html> (Accessed: December, 2012)

- [25] Russel, S. and Norvig, P.: Artificial Intelligence: A Modern Approach Second Edition, Prentice Hall. (2003). 650, 736 – 748
- [26] SANN Overviews – Network Types. From <http://documentation.statsoft.com/STATISTICAHelp.aspx?path=SANN/Overview/SANNOverviewsNetworkTypes> (Accessed: December, 2012)
- [27] Shi, L-C., and Lu., B-L.: EEG-based Vigilance Estimation using Extreme Learning Machines. Neurocomputing, Vol. 102. (2013). 135 – 143
- [28] Şimşekli. U.: Automatic Music Genre Classification Using Bass Lines. 20th IAPR International Conference on Pattern Recognition (ICPR), Istanbul, Turkey, 2010
- [29] Smith, E., et al.: Atkinson & Hilgard's Introduction to Psychology 14th Edition, Thomson Learning Asia. (2003). 32 – 37
- [30] Spring Visual Culture 1B – Neuron Psychologist. From <http://springvisualculture1b.blogspot.com/2010/04/neuron-psychologist.html> (Accessed: December, 2012)
- [31] Tabao, G.: HarmoSegment: HSA-Based Image Segmentation (Unpublished) (2011)
- [32] There's a rock band set up in the Cal Shakes rehearsal hall!. From <http://www.mobypicture.com/user/calshakes/view/9747748> (Accessed: December, 2012)
- [33] Understanding Basic Scales = More Loop Fun. From <http://www.loopblog.net/tutorials/music-theory-tutorials/understanding-basic-scales-more-loop-fun/> (Accessed: December, 2012)
- [34] Wang, D., and Huang, G.B.: Protein Sequence Classification using Extreme Learning Machine. Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Vol.3, (2005) 1406 – 1411
- [35] What to Expect at a Concert. From <http://www.psnj.org/education/concert-guide/what-to-expect-at-a-concert/> (Accessed: December, 2012)
- [36] Yusiong, J.P.: CMSC 170 – Neural Networks. Powerpoint presentation. University of the Philippines Visayas Tacloban College. February 2012.
- [37] Zhang, et al.: Multicategory Classification Using an Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol.4, No.3, (2007) 485 – 495

APPENDIX

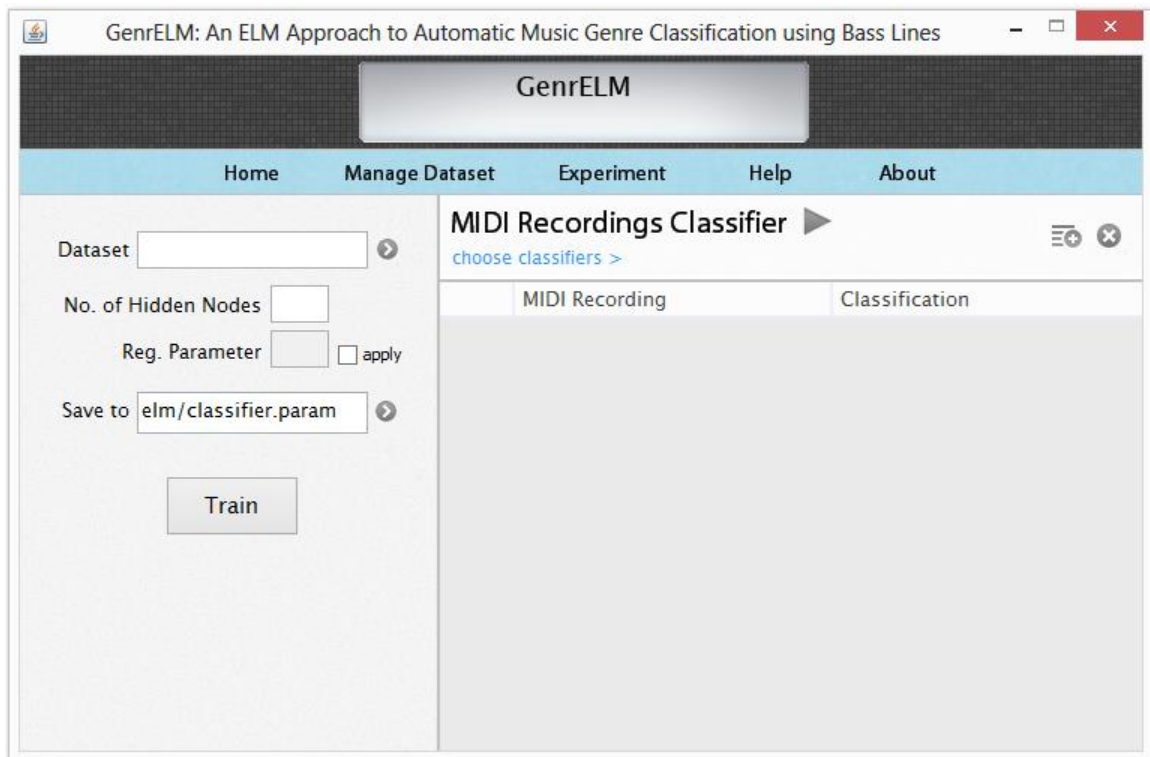
USER'S MANUAL

GenrELM has five sections: the *Home* page, *Manage Dataset* page, *Experiment* page, and the *Help* and *About* pages. A splash screen is first displayed upon running the program, as shown below.



Home Page

Use the *Home* page for testing MIDI files containing bass lines using the proposed approach (*i.e.* hierarchical approach), as well as for training classifiers.



A. Training a Classifier

1. Open training set (dataset) file.

Dataset ➤

2. Input network configuration (*i.e.* number of hidden nodes, regularization coefficient).

No. of Hidden Nodes
Reg. Parameter ☒ apply

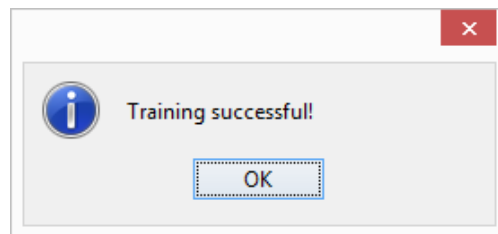
3. Choose save path.

Save to ➤



4. Click “Train” button.




Train

A dialog will pop up to indicate that training has been successful.

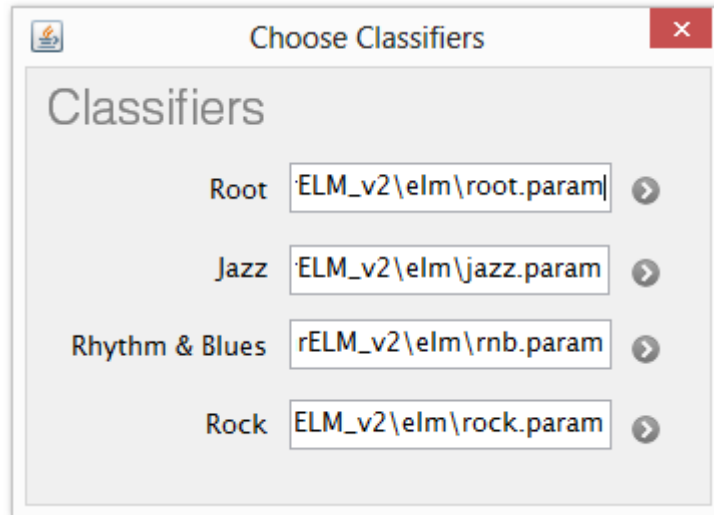



B. Testing MIDI Files

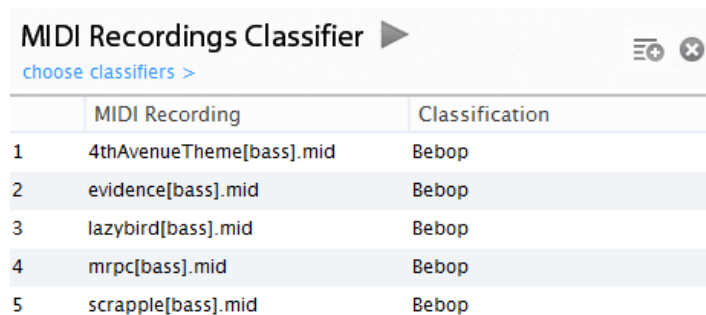
1. Click  to add MIDI files to test. One or more MIDI files may be added at a time. MIDI files should contain bass lines only. The selected MIDI files will be listed as shown in the diagram below. Click  to clear list. To delete a single entry, press “DEL”.

MIDI Recordings Classifier 			 
choose classifiers >			
	MIDI Recording	Classification	
1	4thAvenueTheme[bass].mid		
2	evidence[bass].mid		
3	lazybird[bass].mid		
4	mrpc[bass].mid		
5	scrapple[bass].mid		

2. Click [choose classifiers >](#) to choose your classifiers. GenrELM remembers your selected classifiers, so you won't have to repeat this step if you re-run the program and would like to use your recently selected classifiers.



3. Click  to classify the selected MIDI files. The classification will be displayed next to the MIDI titles as shown in the diagram below.



The 'MIDI Recordings Classifier' window displays the following table:

	MIDI Recording	Classification
1	4thAvenueTheme[bass].mid	Bebop
2	evidence[bass].mid	Bebop
3	lazybird[bass].mid	Bebop
4	mrpc[bass].mid	Bebop
5	scrapple[bass].mid	Bebop

Manage Dataset Page

Use the *Manage Dataset* page to set up your training data.

A. Adding Files for Training

1. Configure the network structure (*i.e.* number of output nodes).

2. Select the genre of the MIDI files to be added to the training set.

3. Click to select MIDI files to be added to the training set. The files will be displayed in a list as shown below. Click to clear the list. To delete an individual entry, press “DEL”.

	MIDI Recording
1	4on6[bass].mid
2	52ndst[bass].mid
3	Bluenboo[bass].mid
4	airegin[bass].mid
5	auprvave[bass].mid

4. Click to add the selected files to the training set. The added files are listed as shown in the diagram below.

Network Structure ☒ 3 output nodes
☐ 9 output nodes

Genre Jazz ☐ Leaf ☒ Root

MIDI Recordings to Add + ×

MIDI Recording

Add

Manage Training Set

⋮ Save

	MIDI Recording	Genre
1	4on6[bass].mid	Jazz
2	52ndst[bass].mid	Jazz
3	Bluenboo[bass].mid	Jazz
4	airegin[bass].mid	Jazz
5	auprvave[bass].mid	Jazz
6	billiesb[bass].mid	Jazz
7	cooker_benson[bass].mid	Jazz
8	dexterity[bass].mid	Jazz
9	donnalee[bass].mid	Jazz
10	grooving[bass].mid	Jazz
11	halfn[bass].mid	Jazz
12	haveiones[bass].mid	Jazz

4. Click Save to save training set. Click ⋮ to clear the list.

Experiment Page

Use the *Experiment* Page for performing tests on the classifiers with given a number of runs. Both training and testing is performed per run.

No. of Output Nodes ☒ 3 ☐ 9

Classification ☒ leaf ☐ root

Jazz

Training Set ?

Hidden Nodes Reg. Coeff. 1

No. of Runs ☐ apply

+ ×

MIDI Recording

Experiment Mode

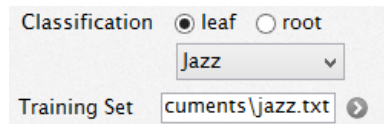
▶

A. Selecting MIDI Files to Test

1. Choose the number of output nodes for experimentation.

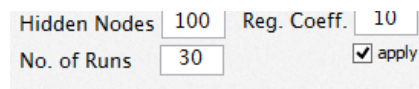
No. of Output Nodes ☒ 3 ☐ 9

2. Enter information about the training set.







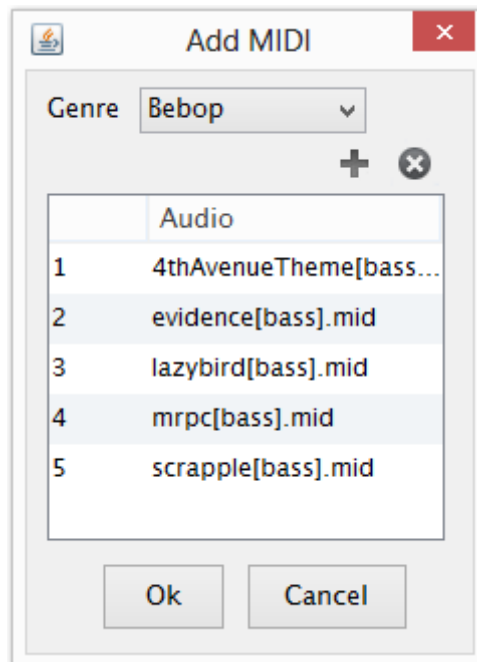
Classification ☒ leaf ☐ root
Jazz
Training Set cuments\jazz.txt

3. Enter the network configuration for the classifier, as well as the number of runs for the experiment.





Hidden Nodes 100 Reg. Coeff. 10
No. of Runs 30 ☒ apply

4. Click  to select MIDI files to be tested. To remove MIDI files, click . A dialog will pop up, asking for the files to be tested with their known genre classification. Choose the genre from the dropdown menu box and click  and  to add and clear files, respectively. To remove a single entry, press “DEL”.




Add MIDI

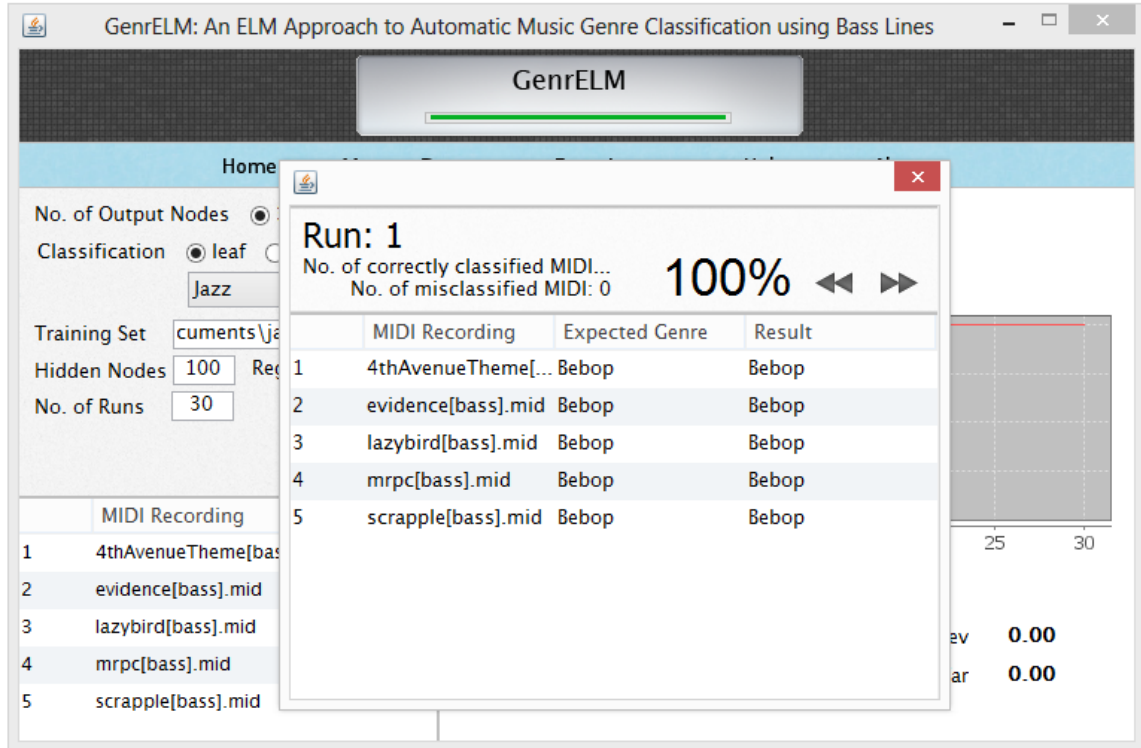
Genre Bebop

	Audio
1	4thAvenueTheme[bass...
2	evidence[bass].mid
3	lazybird[bass].mid
4	mrpc[bass].mid
5	scrapple[bass].mid

Ok Cancel

5. Click  to start. A dialog displaying the results per run will pop up, as shown in the diagram below.



A summary of the experimentation is also shown, as in the diagram below.



Help Page

The *Help* button links to this user's manual.

***About* Page**

The *About* section shows a brief overview of the program, as shown below.

