M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 & ISO 14001:2015 Certified Institution
Thalavapalayam, Karur – 639 113.

CONVOLUTIONAL RECURRENT NEURAL NETWORK FOR MUSIC GENRE CLASSIFICATION ON DIGITAL COMPOSITION

A PROJECT REPORT
(Phase - I)

Submitted by

ABISHEK K K	**(17BEC3005)**
HARIKRISHNAN P	**(17BEC3059)**
MADHAN K	**(17BEC3097)**

in partial fulfillment for the award of degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

M.KUMARASAMY COLLEGE OF ENGINEERING

(An Autonomous Institution, Affiliated to Anna University Chennai), KARUR – 639 113

M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 & ISO 14001:2015 Certified Institution
Thalavapalayam, Karur - 639 113.

BONAFIDE CERTIFICATE

Certified that this project report entitled **"CONVOLUTIONAL RECURRENT NEURAL NETWORK FOR MUSIC GENRE CLASSIFICATION ON DIGITAL COMPOSITION"** is the bonafide work of **"K.K.ABISHEK (17BEC3005), P.HARIKRISHNAN (17BEC3059), K.MADHAN (17BEC3097)"** who carried out the project work during the academic year 2020-2021 under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mr.S.NANDHAGOPAL, M.E.,	Dr.S.JEGADEESAN, M.E., Ph.D.,
SUPERVISOR,	**HEAD OF THE DEPARTMENT,**
Professor,	Professor,
Department of Electronics and	Department of Electronics and
Communication Engineering,	Communication Engineering,
M.Kumarasamy College of Engineering,	M.Kumarasamy College of Engineering,
Thalavapalayam, Karur - 639113.	Thalavapalayam, Karur - 639113.

This project report has been submitted for the **16EC712-Project Work I** End semester viva voce examination held on **.................................**

EXTERNAL EXAMINER **INTERNAL EXAMINER**

INSTITUTION VISION AND MISSION

Vision

To emerge as a leader among the top institutions in the field of technical education.

Mission

M1: Produce smart technocrats with empirical knowledge who can surmount the global challenges.

M2: Create a diverse, fully -engaged, learner -centric campus environment to provide quality education to the students.

M3: Maintain mutually beneficial partnerships with our alumni, industry and professional associations

DEPARTMENT VISION, MISSION, PEO, PO AND PSO

Vision

To empower the Electronics and Communication Engineering students with emerging technologies, professionalism, innovative research and social responsibility.

Mission

M1: Attain the academic excellence through innovative teaching learning process, research areas & laboratories and Consultancy projects.

M2: Inculcate the students in problem solving and lifelong learning ability.

M3: Provide entrepreneurial skills and leadership qualities.

M4: Render the technical knowledge and industrial skills of faculties.

Program Educational Objectives

PEO1: Graduates will have a successful career in academia or industry associated with electronics and communication engineering.

PEO2: Graduates will provide feasible solutions for the challenging problems through comprehensive research and innovation in the allied areas of electronics and communication engineering.

PEO3: Graduates will contribute to the social needs through lifelong learning, practicing professional ethics and leadership quality.

Program Outcomes

PO 1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

PSO1: Applying knowledge in various areas, like Electronics, Communications, Signal processing, VLSI, Embedded systems etc., in the design and implementation of Engineering application.

PSO2: Able to solve complex problems in Electronics and Communication Engineering with analytical and managerial skills either independently or in team using latest hardware and software tools to fulfil the industrial expectations.

Abstract	Matching with POs, PSOs
KEYWORDS	PO1, PO2, PO6, PO7, PO9, PO12, PSO1, PSO2

S.No.	Project Domain	Mapping with POs/PSOs
1.	Antenna	PO1, PO2, PO3, PO6, PO8, PO9,PO10,PO11, PO12, PSO1, PSO2
2.	Cognitive Radio	PO1, PO2, PO3, PO6, PO8, PO9,PO10,PO11, PO12, PSO1, PSO2
3.	Digital Image processing	PO1,PO2,PO3,PO4,PO5,PO6,PO7,PO8,PO9,PO10,PO11,PO12, PSO1, PSO2
4.	Embedded Systems	PO1, PO2, PO3, PO4, PO6, PO7,PO9, PO12, PSO1, PSO2
5.	IOT	PO1, PO2, PO3, PO4,PO5, PO6,PO8,PO9,PO10,PO11, PO12, PSO1, PSO2
6.	PCB	PO1, PO2,PO3,PO5, PO6, PO7,PO8, PO9,PO11, PO12, PSO1, PSO2
7.	VLSI	PO1, PO2,PO3,PO5, PO8, PO9,PO10,PO11, PO12,PSO1, PSO2
8.	Network	PO1, PO2, PO3,PO4,PO5,PO8, PO9, PO10,PO11, PO12, PSO1, PSO2

ACKNOWLEDGEMENT

Our sincere thanks to **Thiru.M.Kumarasamy, Chairman** and **Dr.K.Ramakrishnan, Secretary** of **M.Kumarasamy College of Engineering** for providing extra ordinary infrastructure, which helped us to complete this project in time.

It is a great privilege for us to express our gratitude to **Dr.N.Ramesh Babu, M.E., Ph.D., Principal** for providing us right ambiance to carry out this project work.

We would like to thank **Dr.S.JEGADEESAN, M.E., Ph.D., Associate Professor and Head**, **Department of Electronics and Communication Engineering** for his unwavering moral support and constant encouragement towards the completion of this project work.

We offer our whole hearted thanks to our **Project Supervisor, Mr.S.NANDHAGOPAL, M.E** Professor, Department of Electronics and Communication Engineering for his precious guidance, tremendous supervision, kind co-operation, valuable suggestions and support rendered in making our project to be successful.

We would like to thank our **Project Co-ordinator, Ms.S.AATHILAKSHMI,** Associate Professor, Department of Electronics and Communication Engineering for his kind cooperation and culminating in the successful completion of this project work. We glad to thank all the **Faculty Members** of **Department of Electronics and Communication Engineering** for extending a warm helping hand and valuable suggestions throughout the project. Words are boundless to thank our **Parents** and **Friends** for their motivation to complete this project successfully.

TABLE OF CONTENTS

ABSTRACT

Deep learning algorithm to build deep belief neural networks. The goal is to use it to perform a multi-class classification task of labelling music genres and compare it to that of the neural networks. We expected that deep learning would out-perform how- ever the results we obtained for 2 and 3 class classification turn out to be on par for deep neural networks and the with small data set. By generating more dataset from the original limited music tracks, we see a great classification accuracy improvement in the deep belief neural network and its out- performance than neural networks. Artificial neural networks have great potential in learning complex high-level knowledge from raw inputs thanks to its non-linear representation of the hypothesis function. Companies nowadays use music classification, either to be able to place recommendations to their customers (such as Spotify, Soundcloud) or simply as a product (for example Shazam). Determining music genres is the first step in that direction. Machine Learning techniques have proved to be quite successful in extracting trends and patterns from the large pool of data. The same principles are applied in Music Analysis also.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CNN	Convolutional Neural Network
SVM	Support Vector Machine
KNN	K - Nearest Neighbor

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW

Genre classification is an important task with many real world applications. As the quantity of music being released on a daily basis continues to sky-rocket, especially on internet platforms such as Soundcloud and Spotify – a 2016 number suggests that tens of thousands of songs were released every month on Spotify – the need for accurate meta-data required for database management and search/storage purposes climbs in proportion. Being able to instantly classify songs in any given playlist or library by genre is an important functionality for any music streaming/purchasing service, and the capacity for statistical analysis that correct and complete labeling of music and audio provides is essentially limitless. We implemented a variety of classification algorithms admitting two different types of input. We experimented with a RBF kernel support vector machine, k-nearest neighbors, a basic feed-forward network, and finally an advanced convolutional neural network. For the input to our algorithms, we experimented with both raw amplitude data as well as transformed mel-spectrograms of that raw amplitude data. We then output a predicted genre out of 10 common music genres. We found that converting our raw audio into mel-spectrograms produced better results on all our models, with our convolutional neural network surpassing human accuracy.

Machine learning techniques have been used for music genre classification for decides now. In 2002, G. Tzanetakis and P. Cook [8] used both the mixture of Gaussians model and k-nearest neighbors along with three sets of carefully hand-extracted features representing timbral texture, rhythmic content and pitch content. They achieved 61% accuracy. As a benchmark, human accuracy averages around

70% for this kind of genre classification work [4]. Tzanetakis and Cook used MFCCs, a close cousin of mel-spectrograms, and essentially all work has followed in their footsteps in transforming their data in this manner. In the following years, methods such as support vector machines were also applied to this task, such as in 2003 when C. Xu et al. [9] used multiple layers of SVMs to achieve over 90% accuracy on a dataset containing only four genres. In the past 5-10 years, however, convolutional neural networks have shown to be incredibly accurate music genre classifiers [8] [2] [6], with excellent results reflecting both the complexity provided by having multiple layers and the ability of convolutional layers to effectively identify patterns within images (which is essentially what mel-spectrograms and MFCCs are). These results have far exceeded human capacity for genre classification, with our research finding that current state-of-the-art models perform with an accuracy of around 91% [6] when using the full 30s track length. Many of the papers which implemented CNNs compared their models to other ML techniques, including k-NN, mixture of Gaussians, and SVMs, and CNNs performed favorably in all cases. Therefore we decided to focus our efforts on implementing a high-accuracy CNN, with other models used as a baseline.

Artificial neural networks have great potential in learning complex high level knowledge from raw inputs thanks to its non-linear representation of the hypothesis function. Training and generalizing a neural network with many hidden layers using standard techniques are challenging. According to some related research literatures(2), training deep neural network with the traditional back-propagation algorithm tend to get the network stuck in a local minima. However with the development in pretraining algorithms such as auto-encoder and restricted Boltzmann machine one can achieve better results. Furthermore recently there is a surge in the interests of the deep learning algorithms in both academic and industries. This is partly due to the recent advancements in computational techniques and hardware such as GPU, a lot of deep learning algorithms can now be effectively trained. Many recent findings show that deep neural networks in some tasks outperform most of the traditional classification algorithms such as support vector machine and random forest. What we will be focused on in this project is a branch of automatic speech recognition. Specifically we would like to apply various deep learning algorithms to classify music genres and study their performances. In the first section of the paper, we will set up the problems of interest for the paper and describes the algorithms that we will compare. In section.3 data and data preprocessing are discussed. We then describe the implementation of the algorithms in section.4 and finally in section.5 and 6 I will report the results and discuss what could be done in the future to improve the classifier.

CHAPTER 2
2. AUDIO FEATURES

2.1 MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCCS)

MFCCs are one of the most common feature types used in audio classification of all kinds, having been used in speech and ambient noise recognition as well as music genre classification. Unlike the estimated tempo of a given audio signal, which consists of a single value, ordinarily between 13 and 30 MFCCs are extracted, with the exact number of coefficients a decision made by the researcher. In this project, of the 54 features extracted from the audio file dataset, 30 were MFCCs; thus, it is worth examining the feature type in some detail. MFCCs can be better understood by understanding the process by which they are derived from an audio file, which ordinarily runs as follows (Lyons, 2012): 1. An audio file is divided into small frames (a process known as 'windowing'), each lasting 20-40ms. The resulting vector is a time series that represents the overall audio file. 2. For each frame, an estimation of the power spectrum is calculated, by use of a Fourier Transform. The power spectrum of an audio frame is roughly equivalent to the power present for each of the frequency bands within that frame. 3. A 'Mel-spaced Filterbank' is used on each of the output sets from step 2. The Filterbank is spaced according to the mel scale, to give emphasis to the frequency bands that are most relevant in human perception (humans find it easier to perceive differences in the lower registers, i.e., the bassier frequencies, so the filters are more narrowly spaced in these lower registers than in the higher registers). Each of the filters corresponds to a particular frequency band, and removes all of the values that fall outside this range. The result is a series of 'filterbank energies'; one value per filter. 4. After computing the filterbank energies, their logarithms are calculated. 5. A Discrete Cosine Transform is performed on the logarithms, and approximately half of the resulting values are dropped. The result is a time-series vector of MFCCs: one set of MFCCs for each frame of the audio file. It is further possible to find the mean values for each MFCC of an audio file. For example, to find a signal's mean MFCC5 value, the values of MFCC5 across all windows of the signal would be added up, and this value would then be divided by the number of windows in the signal. This process can be applied for each MFCC to find the mean MFCC values. MFCCs are used in music genre classification to detect timbre (Jensen et al., 2006).

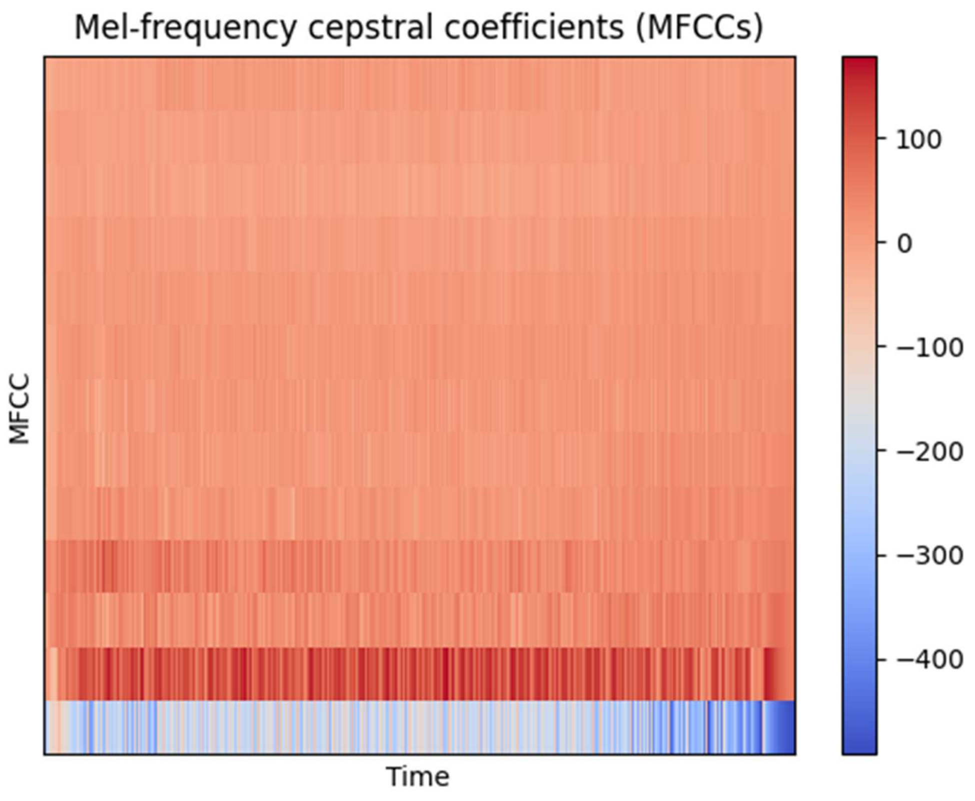

FIGURE 2.1

2.2 ZERO CROSSING RATE (TIMBRAL TEXTURE FEATURE)

A second feature commonly used in music genre recognition is the zero-crossing rate (ZCR). The ZCR of a signal is defined as the rate at which a signal changes from positive to negative or negative to positive (Peeters, 2004, section 4.2). In the context of audio, this refers to the number of times the amplitude of the signal passes through a value of zero, within a given time interval. It is commonly used as a measure of the smoothness of an audio signal (Bäckström, 2019), and has been used as a feature in various forms of audio classification. For example, it has been used to separate speech recordings into periods of 'voiced' speech (sounds made when vowels are spoken) and 'unvoiced' speech (sounds made when consonants are spoken) (Bachu et al., 2008). In the context of music information retrieval, ZCR has been used to identify the presence of percussive sounds (Gouyon, 2000); a factor relevant to genre recognition because percussive sounds are utilized by some genres more than others. Like MFCCs, ZCR is another feature that can be used to understand the timbre of an audio file.

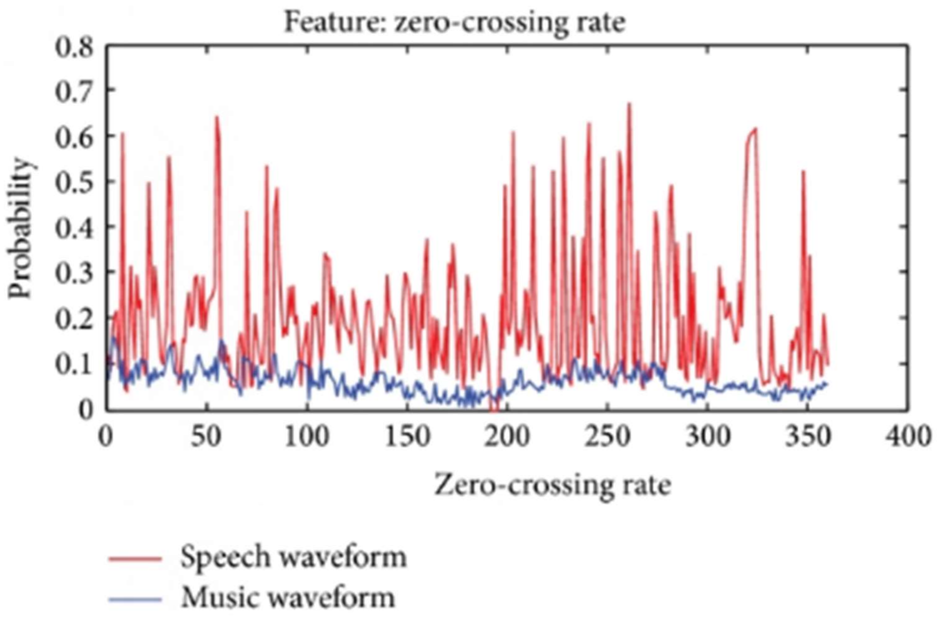

FIGURE 2.2

2.3 SPECTRAL SPREAD (Spectrogram)

'Spectral spread' is understood in the context of 'spectral centroid', which refers to the 'centre of gravity' of the spectrum of a signal (Giannakopoulos & Pikrakis., 2014, 4.4.1 'Spectral Centroid and Spread'). Perceptually, spectral centroid relates to the brightness of a signal's sound. The spectral spread of a signal describes the 'average deviation of the rate map around its centroid' (Two!Ears team, 2015, 'Spectral features'); i.e., the variance. Noisier sounds imply a higher spectral spread, whereas signals with a spectrum 'tightly concentrated' around the centroid will have a lower spectral spread (Giannakopoulos & Pikrakis, 2014, 4.4.1 'Spectral Centroid and Spread'). Like MFCCs and ZCR, spectral spread is used to understand the timbre of an audio file, and is another feature that has been used in music genre recognition. For example, Giannakopoulos et al found that spectrograms of electronic music excerpts are typically spread more widely around their centroid than are excerpts of jazz (ibid.).

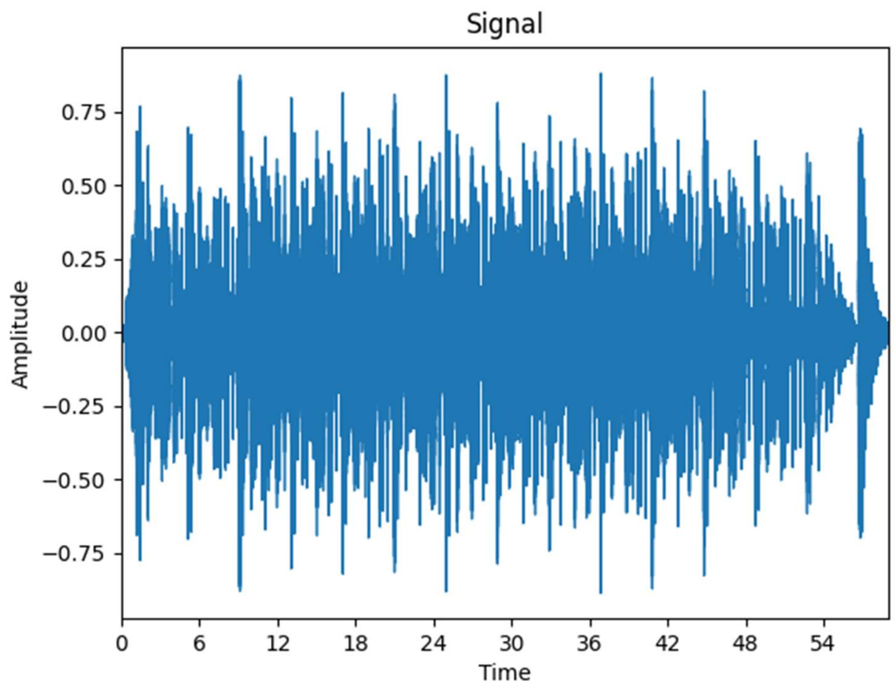

FIGURE 2.3

2.4 FAST FOURIER TRANSFORM

A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. The DFT is obtained by decomposing a sequence of values into components of different frequencies.[1] This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. As a result, it manages to reduce the complexity of computing the DFT from , which arises if one simply applies the definition of DFT, to , where is the data size. The difference in speed can be enormous, especially for long data sets where *N* may be in the thousands or millions. In the presence of round-off error, many FFT algorithms are much more accurate than evaluating the DFT definition directly or indirectly. There are many different FFT algorithms based on a wide range of published theories, from simple complex-number arithmetic to group theory and number theory.

Fast Fourier transforms are widely used for applications in engineering, music, science, and mathematics. The basic ideas were popularized in 1965, but some algorithms had been derived as early as 1805. In 1994, Gilbert Strang described the FFT as "the most important numerical algorithm of our lifetime", and it was included in Top 10 Algorithms of 20th Century by the IEEE magazine Computing in Science & Engineering.

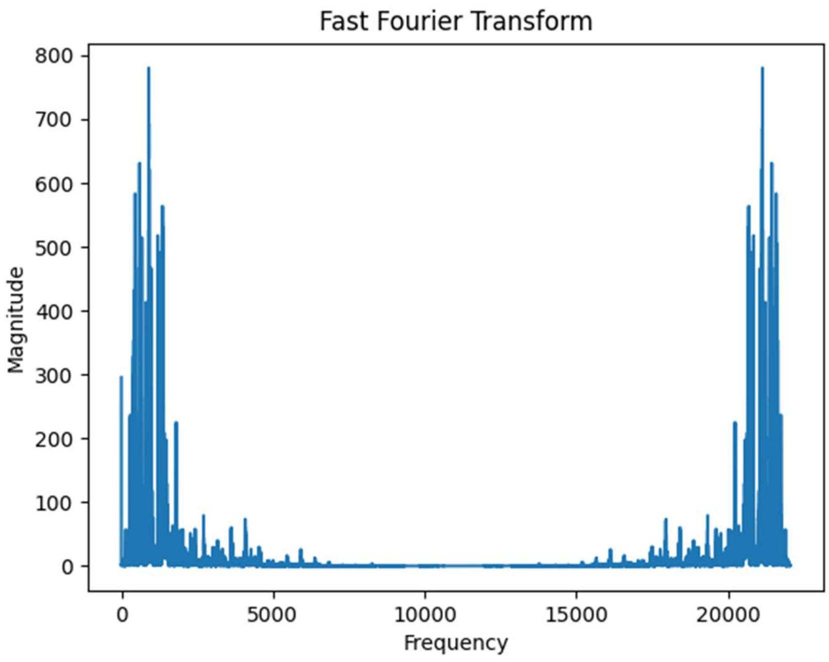

FIGURE 2.4.1

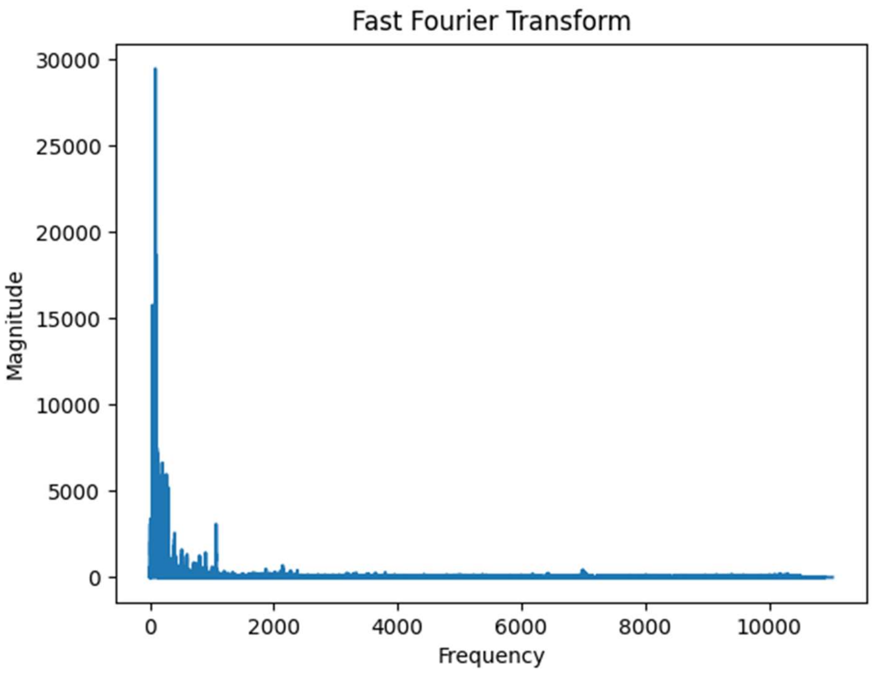

FIGURE 2.4.2

2.5 SHORT FOURIER TRANSFORM

The Short-time Fourier transform (STFT), is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time.[1] In practice, the procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the Fourier transform separately on each shorter segment. This reveals the Fourier spectrum on each shorter segment. One then usually plots the changing spectra as a function of time, known as a spectrogram or waterfall plot, such as commonly used in Software Defined Radio based spectrum displays. Full bandwidth displays covering the whole range of an SDR commonly use FFTs with 2^24 points on desktop computers. STFTs as well as standard Fourier transforms and other tools are frequently used to analyze music. The spectrogram can, for example, show frequency on the horizontal axis, with the lowest frequencies at left, and the highest at the right. The height of each bar (augmented by color) represents the amplitude of the frequencies within that band. The depth dimension represents time, where each new bar was a separate distinct transform. Audio engineers use this kind of visual to gain information about an audio sample, for example, to locate the frequencies of specific noises (especially when used with greater frequency resolution) or to find frequencies which may be more or less resonant in the space where the signal was recorded. This information can be used for equalization or tuning other audio effects.

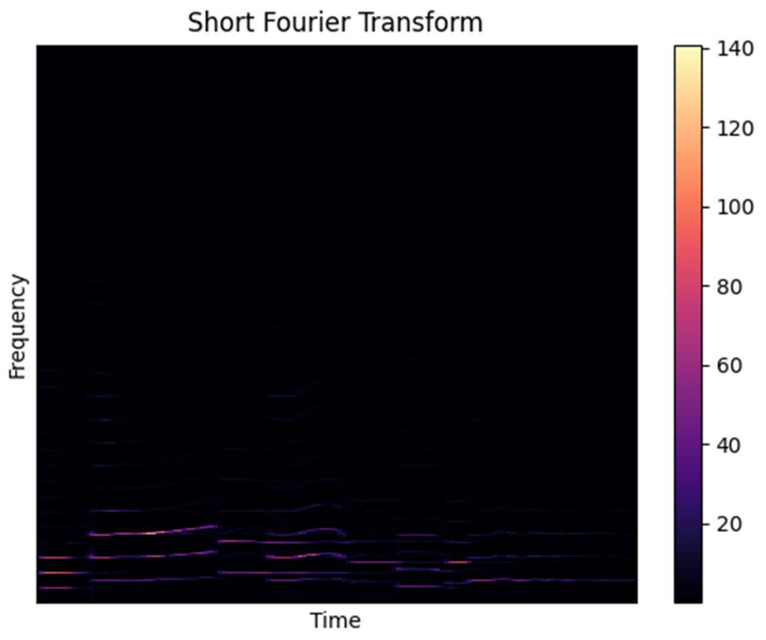

FIGURE 2.5.1

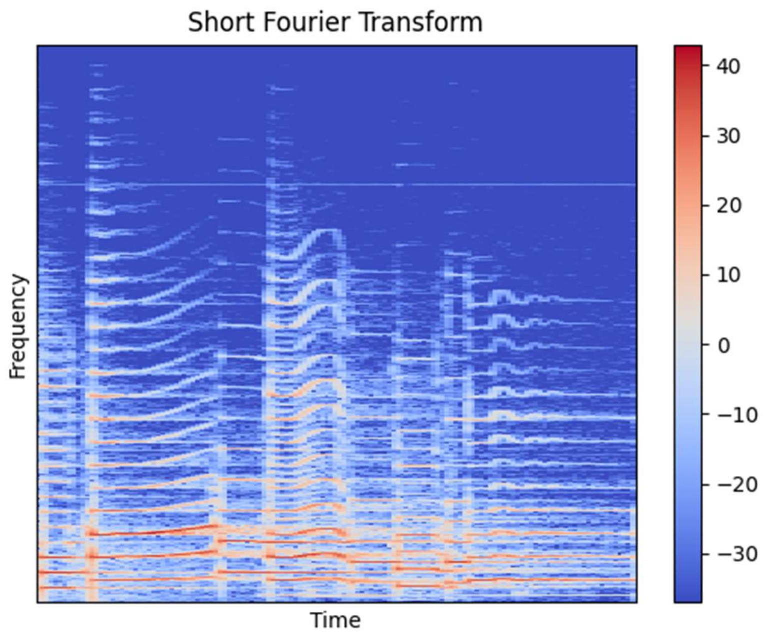

FIGURE 2.5.2

CHAPTER 3
3. ALGORITHMS

The remainder of this chapter will look more closely at five machine learning algorithms that are frequently used for classification problems, and will include discussion of how such methods can be applied to the task of music genre recognition.

3.1 ARTIFICIAL NEURAL NETWORK

We have choosen Convolutional Neural Network especially for this project. Artificial neural networks are a popular form of machine learning classifier, consisting of layers of connected nodes that transform input data to some kind of output (dependent upon the specific implementation). In the context of classification, neural networks learn models from training data, in order to predict the class of unseen data samples. Various kinds of neural network exist, including fully-connected, convolutional, and recurrent, and networks can be structured with varying numbers of nodes and layers according to design choices made by the engineer. A number of strategies exist for combating overfitting and reducing model complexity, and, when architected correctly, neural networks have proven to be especially effective for classification problems. The exact structure of the network used for this project will be provided in the 'models' section of this report. A range of approaches can be taken toward neural network input. The shape of the first layer of the network will depend upon choices made in the feature engineering stage of the project. For example, in image recognition, the number of 'input nodes' will usually equal the number of pixels of each image of the dataset. Each pixel is represented by a number, and, if structured correctly, the network can be trained to learn 'features' of the images, e.g. the curve of a '2' in handwriting recognition. However, when features have been manually selected from the data, as is the case for this project, the number of input nodes is determined by the number of features extracted in this way. Audio data can be described as 'time-series data', in that it describes sonic events that take place over time. Because of this, there exists the possibility of representing audio files as time-series vectors, in terms of features such as MFCCs. Recurrent Neural Networks make use of loops and are able to 'remember' information between stages of the network's processing. This makes them especially suited to sequential and time-based data (Karpathy, 2015, 'Recurrent Neural Networks'). Consequently, RNNs have been used for audio classification problems such as speech recognition (Olah, 2015). The use of RNNs for music genre classification presents an interesting and potentially fruitful opportunity, especially the use of LSTMs (Irvin, Chartock & Hollander, 2016). However, due to time and resource constraints, use of RNNs was ultimately deemed to be out of scope for this particular project, and other architectures were experimented with instead. in neural networks, convolutional neural network (convnets or cnns) is one of the main categories to do images recognition, images classifications. objects detections, recognition faces etc., are some of the areas where cnns are widely used.

3.2 SUPPORT-VECTOR MACHINE

SVMs are another type of supervised learning machine learning algorithm. In classification problems, data points of the same class will oftentimes cluster together on graphs that plot their feature values. In order to make predictions for new data points, a 'n-1 dimensional hyperplane' can be used to carve up the graph, where n = the number of features. A new data point falling on one side of the hyperplane will be predicted to be of one class; a data point falling to the other side will be predicted another. SVMs attempt to find the optimal hyperplane for a given dataset, ordinarily defined as the hyperplane that maximizes the margin between the classes. SVMs can be initialized with a range of different kernel functions, which create different types of data divisors. A linear kernel function attempts to divide data points with a linear hyperplane, whereas a 'kernel trick' can be used to create polynomial decision boundaries, by effectively mapping the data points into higherdimensional space so that a dividing hyperplane can be drawn. Depending on how disparate two given classes are, the clusters may be 'far apart' on the graph, or there may be some degree of overlap, and the success of a support vector machine implementation will partly depend on the ease with which the classes can be separated. If, as has been argued above, different musical genres produce distinct average feature values, then support vector machines may prove promising for music genre classification, as the clusters of each genre could be separated with hyperplane divisors relatively easily.

3.3 K – NEAREST NEIGHBOURS

The K-Nearest Neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

In the case of classification and regression, we saw that choosing the right K for our data is done by trying several Ks and picking the one that works best.

At scale, this would look like recommending products on Amazon, articles on Medium, movies on Netflix, or videos on YouTube. Although, we can be certain they all use more efficient means of making recommendations due to the enormous volume of data they process.

However, we could replicate one of these recommender systems on a smaller scale using what we have learned here in this article. Let us build the core of a movies recommender system.

CHAPTER 4

4. PROPOSED SYSTEM

The Main quantitative metric which we used to judge our models is accuracy (that is, percentage of predicted labels which matched their true labels), and our main way of visualizing the performance of our best model is through the confusion matrices as seen . Because the labeling was uniformly distributed on our data, cross-validation, and test sets, these confusion matrices offer not only a way to visualize our data, but more specific information than precision and recall values offer. We selected our hyperparameters based of empirical results and industry standards. For instance, choosing 4 by 4 window for our first convolution window was a result of seeing a similar window size work in other academic results, and then fine tuning to meet our data-specific needs. We chose to use Adam optimization for a few reasons. Working with audio time-series data over 2 dimensions cause sparse gradient problems, similar to those often encountered in natural language or computer vision problems. We also felt our data was somewhat noisy and messy. Adam mitigates the sparse gradient problem by maintaining a per-parameter learning rate, and mitigates the noise problem by basing updates on a weighted average of recent updates (momentum). With Adam our models trained more quickly and didn't plateau as early. The clearest trend identifiable here is the dramatic jump in performance after we moved from using raw audio to mel-spectrograms. We saw a substantial improvement in accuracy on all four of our models after converting our data to this form, which suggests that there is essentially no benefit to looking directly at raw amplitude data. While it is true that converting to mel-spectrograms takes a little bit of time, especially with our high number of training examples, this pre-processing step can be pre-computed and stored in a file format such as .npz for quick access across all models. Additionally, mel-spectrograms are essentially images, which provides an human-accessible way to Confusion matrix for CNN predictions.

visualize our data and to think about what our neural networks may be classifying on. In other words, there is essentially no downside to switching to mel-spectrograms. We also see that all four of our models struggle with over-fitting. We spent the most time trying to mitigate this issue on the CNN. To do so, we introduced three methods. We played around with a combination of batch normalization, dropout layers, and L2 regularization. We found some difficulty in using this, as allowing the model to over-fit actually increased our accuracy. While we could bring the training accuracy and the test accuracy to within .05 of each other, this would result in poor model performance. Thus we accepted our over-fitting issue. Looking more closely at our confusion matrix, we see that our CNN struggled most with the rock genre. It only managed to correctly classify 50% of rock audio as rock, labeling the others as mainly country or blues. Additionally, it incorrectly classified some country, as well as a small fraction of blues and reggae, as rock music. While it's not all that surprising that rock was a challenging genre – a qualitative inspection of rock mel-spectrograms implies that many rock music excerpts lack the easily visible beats that other genres such as hip-hop and disco possess, while our personal experience with rock music vis a vis the other genres tells us that rock is also missing distinctive traits such as high-register vocals (pop) or easily audible piano (classical or jazz). Additionally, rock is a genre that both encapsulates many different styles (light rock, hard rock, progressive rock, indie rock, new wave, etc.) and heavily influences many other derivative genres. However, we were surprised that rock and country were so easily confused, as opposed to rock and metal, which would seem to rely on more similar instrumentation and tempo.

4.1 BLOCK DIAGRAM

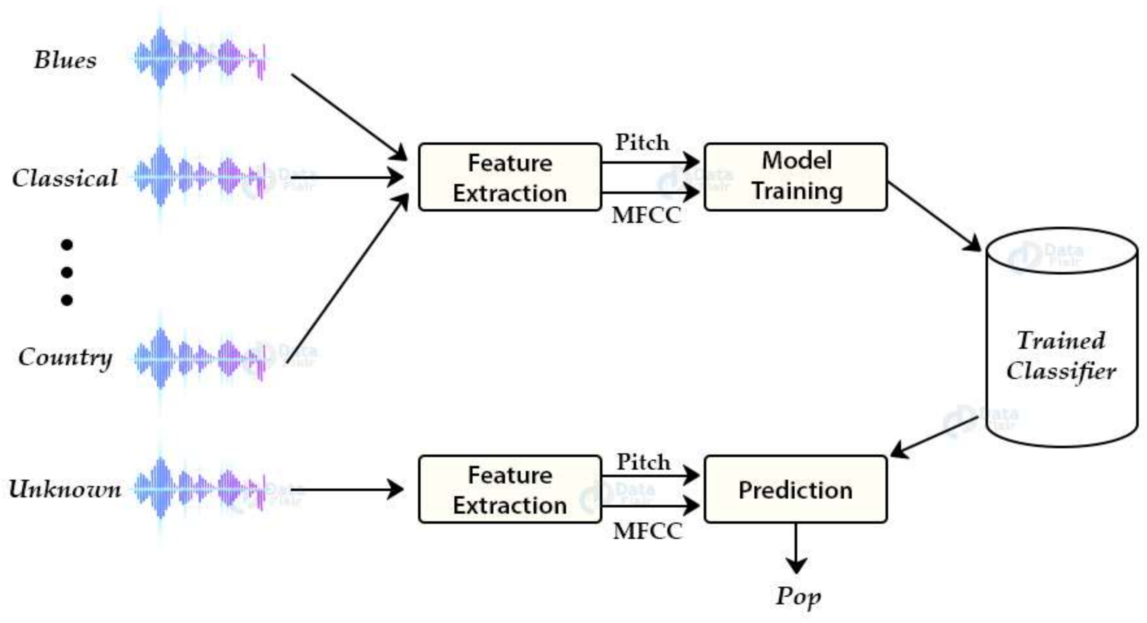

FIGURE 4.1

4.2 ADVANTAGES

- The extracted signal and data are used for many applications
- It can be used in Noise Cancelling Headphones
- Used to classify Unknown waves generated from satellites
- Provide improved accuracy rate

4.3 PROGRAM (CODING)

We have used Python for this Specific Project and used Flask framework to implement it as a webpage. lask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Librosa is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.
Python_speech_features is the library which provides common speech features for ASR including MFCCs and filterbank energies.

4.3.1 IMPORTS

```
from flask import Flask,redirect,render_template,request
from python_speech_features import mfcc
import scipy.io.wavfile as wav
import numpy as np
from matplotlib import cm
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import librosa,librosa.display
from tempfile import TemporaryFile
import os
import pickle
import random
import shutil
import operator
import math
```

4.3.2 AUDIO FEATURE PLOT

```
#audio signal
signal,sr=librosa.load(file,sr=22050)
librosa.display.waveplot(signal,sr=sr)
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title('Signal')
```

```
a="static/data/"+str(random.randint(215500,352400))+'a'
plt.savefig(a)
plt.close()

#Fast Fourier Transform
fft=np.fft.fft(signal)
magnitude=np.abs(fft)
frequency=np.linspace(0,sr,len(magnitude))
left_frequency=frequency[:int(len(frequency)/2)]
left_magnitude=magnitude[:int(len(frequency)/2)]
plt.plot(left_frequency,left_magnitude)
plt.xlabel('Frequency')
plt.ylabel('Magnitude')
plt.title('Fast Fourier Transform')
b="static/data/"+str(random.randint(215500,352400))+'b'
plt.savefig(b)
plt.close()



#short fourier transform
n_fft=2048
hop_length=512
stft=librosa.core.stft(signal,hop_length=hop_length,n_fft=n_fft)
spectrogram = np.abs(stft)
#log_spectrogram=librosa.amplitude_to_db(spectrogram)
librosa.display.specshow(spectrogram,sr=sr,hop_length=hop_length)
```

```
 plt.xlabel("Time")
 plt.ylabel("Frequency")
 plt.colorbar()
 plt.title('Short Fourier Transform')
 c="static/data/"+str(random.randint(215500,352400))+'c'
 plt.savefig(c)
 plt.close()

#MFCC
MFCCs=librosa.feature.mfcc(signal,n_fft=n_fft,
hop_length=hop_length,n_mfcc=13)
 librosa.display.specshow(MFCCs,sr=sr,hop_length=hop_length)
 plt.xlabel("Time")
 plt.ylabel("MFCC")
 plt.colorbar()
 plt.title('Mel-frequency cepstral coefficients (MFCCs)')
 d="static/data/"+str(random.randint(215500,352400))+'d'
 plt.savefig(d)
 plt.close()
```

4.3.3 FUNCTION TO FIND NEAREST NEIGHBOUR

```
# function to get the distance between feature vecotrs and find neighbors
        def getNeighbors(trainingSet, instance, k):
        distances = []
        for x in range(len(trainingSet)):
```

```
  dist = distance(trainingSet[x], instance, k) +                    distance(instance,
  trainingSet[x], k)
          distances.append((trainingSet[x][2], dist))
          distances.sort(key=operator.itemgetter(1))
          neighbors = []
          for x in range(k):
              neighbors.append(distances[x][0])
                  return neighbors

# identify the class of the instance
def nearestClass(neighbors):
    classVote = {}
    for x in range(len(neighbors)):
        response = neighbors[x]
        if response in classVote:
            classVote[response] += 1
        else:
            classVote[response] = 1
            sorter          =          sorted(classVote.items(),          key          =
            operator.itemgetter(1),reverse=True)
    return sorter[0][0]
```

4.3.4 MODAL EVALUATION

```
# function to evaluate the model
def getAccuracy(testSet, prediction):
    correct = 0
```

```
  for x in range(len(testSet)):
      if testSet[x][-1] == predictions[x]:
          correct += 1
 return (1.0 * correct) / len(testSet)
```

4.3.5 EXTRACT FEATURES FROM DATASET

```
directory = "__path_to_dataset__"
f= open("my.dat" ,'wb')
i=0
for folder in os.listdir(directory):
i+=1
if i==11 :
break
for file in os.listdir(directory+folder):
(rate,sig) = wav.read(directory+folder+"/"+file)
mfcc_feat = mfcc(sig,rate ,winlen=0.020, appendEnergy = False)
 covariance = np.cov(np.matrix.transpose(mfcc_feat))
mean_matrix = mfcc_feat.mean(0)
feature = (mean_matrix , covariance , i)
pickle.dump(feature , f)
f.close()
```

4.3.6 SPLIT EXTRACTED DATA FOR TRAIN AND TEST

```
   # Split the dataset into training and testing sets respectively
         dataset = []


def loadDataset(filename, split, trSet, teSet):
    with open('static/TrainedModel/my.dat', 'rb') as f:
        while True:
            try:
                dataset.append(pickle.load(f))
            except EOFError:
                f.close()
                break
    for x in range(len(dataset)):
        if random.random() < split:
            trSet.append(dataset[x])
        else:
            teSet.append(dataset[x])


trainingSet = []
testSet = []
loadDataset('static/TrainedModel/my.dat', 0.66, trainingSet, testSet)
```

4.3.7 MAKE PREDICTION USING KNN

```
    distance =0
    mm1 = instance1[0]
    cm1 = instance1[1]
    mm2 = instance2[0]
    cm2 = instance2[1]
    distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))
  distance+=(np.dot(np.dot((mm2-mm1).transpose() ,
  np.linalg.inv(cm2)) , mm2-mm1 ))
  distance+= np.log(np.linalg.det(cm2)) –
   np.log(np.linalg.det(cm1))
    distance-= k
    return distance

# making predictions using KNN
leng = len(testSet)
predictions = []
for x in range(leng):
    predictions.append(nearestClass(getNeighbors(trainingSet, testSet[x], 5)))

accuracy1 = getAccuracy(testSet, predictions)
#print(accuracy1)
```

4.3.8 TEST MODEL WITH USER INPUT

```
test_dir = "static/data/"

 test_file = test_dir + "test.wav"

 (rate, sig) = wav.read(test_file)
 mfcc_feat = mfcc(sig, rate, winlen=0.020, appendEnergy=False)
 covariance = np.cov(np.matrix.transpose(mfcc_feat))
 mean_matrix = mfcc_feat.mean(0)

 feature = (mean_matrix, covariance, 0)

 results={1: 'blues', 2: 'classical', 3: 'country',
             4: 'disco', 5: 'hiphop', 6: 'jazz', 7: 'metal',
            8: 'pop', 9: 'reggae', 10: 'rock'}
 pred = nearestClass(getNeighbors(dataset, feature, 5))
print(results[pred])
```

4.3.9 ENTIRE SCRIPT

```
from flask import Flask,redirect,render_template,request
from python_speech_features import mfcc
import scipy.io.wavfile as wav
import numpy as np
from matplotlib import cm
import matplotlib
matplotlib.use('Agg')
```

```
import matplotlib.pyplot as plt
import librosa,librosa.display

from tempfile import TemporaryFile

import os
import pickle
import random
import shutil
import operator

import math

UPLOAD_FOLDER = 'static/data/'

app=Flask(__name__)

app.config['UPLOAD_FOLDER']=UPLOAD_FOLDER



@app.route('/')
def index():
    try:
        shutil.rmtree('static/data')
    except:
        os.makedirs('static/data')
    os.makedirs('static/data')
    return render_template("index.html")

@app.route('/upload',methods=['GET','POST'])
def main():
    if request.method=='POST':
        file = request.files['file']
        filename = "test.wav"
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

        file="static/data/test.wav"

        #audio signal
        signal,sr=librosa.load(file,sr=22050)
        librosa.display.waveplot(signal,sr=sr)
        plt.xlabel("Time")
        plt.ylabel("Amplitude")
        plt.title('Signal')
        a="static/data/"+str(random.randint(215500,352400))+'a'
        plt.savefig(a)
        plt.close()

        #Fast Fourier Transform
        fft=np.fft.fft(signal)

        magnitude=np.abs(fft)
        frequency=np.linspace(0,sr,len(magnitude))
```

```
left_frequency=frequency[:int(len(frequency)/2)]
left_magnitude=magnitude[:int(len(frequency)/2)]

plt.plot(left_frequency,left_magnitude)
plt.xlabel('Frequency')
plt.ylabel('Magnitude')
plt.title('Fast Fourier Transform')
b="static/data/"+str(random.randint(215500,352400))+'b'
plt.savefig(b)
plt.close()

#short fourier transform

n_fft=2048
hop_length=512

stft=librosa.core.stft(signal,hop_length=hop_length,n_fft=n_fft)
spectrogram = np.abs(stft)

#log_spectrogram=librosa.amplitude_to_db(spectrogram)

librosa.display.specshow(spectrogram,sr=sr,hop_length=hop_length)

plt.xlabel("Time")
plt.ylabel("Frequency")
plt.colorbar()
plt.title('Short Fourier Transform')
```

```
        c="static/data/"+str(random.randint(215500,352400))+'c'
        plt.savefig(c)
        plt.close()
        #MFCC

MFCCs=librosa.feature.mfcc(signal,n_fft=n_fft,hop_length=hop_length,n_
mfcc=13)
        librosa.display.specshow(MFCCs,sr=sr,hop_length=hop_length)
        plt.xlabel("Time")
        plt.ylabel("MFCC")
        plt.colorbar()
        plt.title('Mel-frequency cepstral coefficients (MFCCs)')
        d="static/data/"+str(random.randint(215500,352400))+'d'
        plt.savefig(d)
        plt.close()
        # function to get the distance between feature vecotrs and find neighbors
        def getNeighbors(trainingSet, instance, k):
            distances = []
            for x in range(len(trainingSet)):
                dist = distance(trainingSet[x], instance, k) + distance(instance,
trainingSet[x], k)
                distances.append((trainingSet[x][2], dist))
            distances.sort(key=operator.itemgetter(1))
            neighbors = []
            for x in range(k):
                neighbors.append(distances[x][0])
            return neighbors
```

```
# identify the class of the instance
def nearestClass(neighbors):
    classVote = {}

    for x in range(len(neighbors)):
        response = neighbors[x]
        if response in classVote:
            classVote[response] += 1
        else:
            classVote[response] = 1

    sorter = sorted(classVote.items(), key = operator.itemgetter(1),
reverse=True)

    return sorter[0][0]

# function to evaluate the model
def getAccuracy(testSet, prediction):
    correct = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == predictions[x]:
            correct += 1

    return (1.0 * correct) / len(testSet)

# Split the dataset into training and testing sets respectively
dataset = []
```

```
def loadDataset(filename, split, trSet, teSet):
    with open('static/TrainedModel/my.dat', 'rb') as f:
        while True:
            try:
                dataset.append(pickle.load(f))
            except EOFError:
                f.close()
                break
    for x in range(len(dataset)):
        if random.random() < split:
            trSet.append(dataset[x])
        else:
            teSet.append(dataset[x])

trainingSet = []
testSet = []
loadDataset('static/TrainedModel/my.dat', 0.66, trainingSet, testSet)

def distance(instance1 , instance2 , k ):
    distance =0
    mm1 = instance1[0]
    cm1 = instance1[1]
    mm2 = instance2[0]
    cm2 = instance2[1]
    distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))
```

```
        distance+=(np.dot(np.dot((mm2-mm1).transpose()                    ,
np.linalg.inv(cm2)) , mm2-mm1 ))
        distance+= np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))
        distance-= k
        return distance

    # making predictions using KNN
    leng = len(testSet)
    predictions = []
    for x in range(leng):
        predictions.append(nearestClass(getNeighbors(trainingSet,
testSet[x], 5)))

    accuracy1 = getAccuracy(testSet, predictions)
    #print(accuracy1)

    test_dir = "static/data/"

    test_file = test_dir + "test.wav"

    (rate, sig) = wav.read(test_file)
    mfcc_feat = mfcc(sig, rate, winlen=0.020, appendEnergy=False)
    covariance = np.cov(np.matrix.transpose(mfcc_feat))
    mean_matrix = mfcc_feat.mean(0)

    feature = (mean_matrix, covariance, 0)
```

```
        results={1: 'blues', 2: 'classical', 3: 'country', 4: 'disco', 5: 'hiphop', 6:
'jazz', 7: 'metal', 8: 'pop', 9: 'reggae', 10: 'rock'}

        pred = nearestClass(getNeighbors(dataset, feature, 5))
        #print(results[pred])
        img='static/images/'+results[pred]+'.jpeg'

        return
render_template("result.html",resultimg=img,result=results[pred].upper(),ac
curacy=round((accuracy1*100),4),a=a+".png",b=b+".png",c=c+".png",d=d+
".png")

    else:
        return redirect('/')


if __name__ == '__main__':
    app.run(debug=True, host='localhost',port=80)
```

4.4 GTZAN Data Set Features

The GTZAN genre collection dataset was collected in 2000-2001. It consists of 1000 audio files each having 30 seconds duration. There are 10 classes (10 music genres) each containing 100 audio tracks. Each track is in .wav format. It contains audio files of the following 10 genres:

- Blues
- Classical
- Country
- Disco
- Hiphop
- Jazz
- Metal
- Pop
- Reggae
- Rock

CHAPTER 5

5. CONCLUSION AND FUTURE ENHANCEMENT

In this project, we overview the various techniques and algorithms are proposed for music genre classification with high efficiency. Across all models, using frequency based mel-spectrograms produced higher accuracy results. Whereas amplitude only provides information on intensity, or how “loud" a sound is, the frequency distribution over time provides information on the content of the sound. Additionally, mel-spectrograms are visual, and CNNs work better with pictures. The CNN performed the best, as we expected. It took the longest time to train as well, but the increase in accuracy justifies the extra computation cost. However, we were surprised to see the similarity in accuracy between the KNN, SVM, and feed-forward neural network. In the future, we hope to experiment with other types of deep learning methods, given they performed the best. Given that this is time series data, some sort of RNN model may work well (GRU, LSTM, for example). We are also curious about generative aspects of this project, including some sort of genre conversion (in the same vein as generative adversarial networks which repaint photos in the style of Van Gogh, but for specifically for music). Additionally, we suspect that we may have opportunities for transfer learning, for example in classifying music by artist or by decade.

5.1 FUTURE ENHANCEMENT

We can implement this model to classify multiple genre in a particular song and classify based on the majority of genre and also planning to increase the timing of the test set or uploaded music.

REFERENCES

1.G. Tzanetakis and P. Cook, "Musical genre classification of audio signals", IEEE Transactions on Speech and Audio Processing, vol. 10, pp. 293-302, 2002.

2.Max Chevalier and Christine Julien, Music recommendation in collaborative and social information retrieval and access, pp. 212, 2009.

3.T. Tolonen and M. Karjalainen, "A computationally efficient multipitch analysis model", IEEE Transactions Speech Audio Processing, vol. 8, pp. 708-716, Nov. 2000.

4.M. Tamura, T. Masuko, K. Tokuda and T. Kobayashi, "Adaptation of pitch and spectrum for HMM-based speech synthesis using MLLR", IEEE International Conference on Acoustics Speech and Signal Processing, vol. 1, pp. 805-808, May 2010.

5.M. Dash and H. Liu, Feature selection for classification Intelligent Data Analysis, vol. 1, no. 4, pp. 131-156, 1997.

6.D. Perrort and G. Jerdigen, "An exploration of factors in the identification of musical style", proceeding of the 1999 society for music perception and cognition, pp. 88.

7.K.,D. Martin, E.D. Scheirer and B., L. Vercoe, "Musical content analysis through models of audition", Proceedings of the 1998 ACM Multimedia Workshop on Content-Based Processing of Music.

8.Beatriz A. Garro and Roberto A. Vazquez, "Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms", Computational Intelligence and Neuroscience, 2015.

9.Bing Xue, Classification Algorithms in Particle Swarm Optimisation for Feature Selection in Classification, pp. 21-25, 2014.

10.David Opitz and Richard Maclin, "Popular Ensemble Methods: An Empirical Study", Journal of Artificial Intelligence Research, vol. 11, pp. 169-198, 1999.

11.Kristína Machová, František Barčák and Peter Bednár, "A Bagging Method using Decision Trees in the Role of Base Classifiers", Acta Polytechnica Hungarica, vol. 3, no. 2, 2006.

12.Maneesh Singhal and Ramashankar Sharma, "Optimization of Naïve Bayes Data Mining Classification Algorithm", International journal for reseach in applied science and engineering technology (IJRASET), vol. 2, no. VIII, August 2014.

13.L. Oliveira, R. Sabourin, F. Bortolozzi and C. Suen, "Feature selection using multi-objective genetic algorithms for handwritten digit recognition", 16th International Conference on Pattern Recognition (ICPR 02), vol. 1, pp. 568-571, 2002.

14.Jun Yang, Fa-Long Luo and Arye Nehorai, "Spectral contrast enhancement: Algorithms and comparisons", Speech Communication, vol. 39, pp. 33-46, 2003.

15.Jayanta Kumar Basu, Debnath Bhattacharyya and Tai-hoon Kim, "Use of Artificial Neural Network in Pattern Recognition", International Journal of Software Engineering and Its Applications, vol. 4, no. 2, pp. 24-34, April 2010.

16.Piero Baraldi, Enrico Zio and Davide Roverso, "Feature Selection For Nuclear Transient Diagnostics", OECD Halden Reactor Project, January 2007.